Commodore

Cena 12 tys. zł
nr indeksu 355275

2'93

KEFIR

Miesięcznik Użytkowników Komputerów C-64 i Amiga

MOTOROLA 68020

POCKET RADIO

GRY...GRY...

Commodore



nr indeksu 355275

Wydawca:
KEBAB - sp. z o.o.
ul. Wojciechowskiego 28
PL - 71-476 Szczecin
telefon (091) 776-74

Redaguje kolegium
w składzie:
Krzysztof Kobus
Patryk Łogiewa
Grzegorz Mikuła
Krzysztof Moroń
Marcin Orłowski
Zbigniew Piotrowicz
Miłosław Smyk
Paweł Sołtysiński

Redaktor naczelny:
Patryk Łogiewa

Szef działu AMIGA:
Krzysztof Kobus
tel. (091) 525-336

Szef działu C-64:
Paweł Sołtysiński
tel. (091) 776-74

Kontakt elektroniczny:
KOBUSKPS@PLSZUS11

Redakcja nie zwraca nie zamówionych materiałów oraz zastrzega sobie prawo wprowadzania zmian w otrzymanych rękopisach.

Wydawca nie odpowiada za treść zamieszczanych ogłoszeń.

Projekt okładki:
Tomasz Kuczyński

Commodore



PRENUMERATA

Każdy egzemplarz zakupiony bezpośrednio u nas kosztuje odpowiednio:
numery: 1; 2/3; 4; 5; 6 - **9,5 tys. zł**
numery: 7/8; 9; 10 oraz następne - **11 tys. zł.**
(UWAGA! - nakład numeru 5"92 jest wyczerpany).

Oznacza to, że można zamówić numery zaległe jak też zaprenumerować jeszcze nie wydane. Odbywa się to tylko poprzez dokonanie odpowiedniej wpłaty na nasze konto. Na odwrocie każdego odcinka kuponu wpłaty należy dokładnie napisać, których numerów wpłata dotyczy.

W przypadku prenumeraty można zamawiać numery do końca aktualnego okresu "małej prenumeraty". W roku 1993 "mała prenumerata" będzie obejmować cztery egzemplarze, z podziałem roku na trzy okresy:

I - numery 1, 2, 3, 4 - 4 egz. x 11 tys. = 44 tys. zł
II - numery 5, 6, 7, 8 - 4 egz. x 11 tys. = 44 tys. zł itd.
(jeśli do tego czasu cena nie ulegnie zmianie).

Prosimy nie przysyłać do redakcji dowodów wpłat.

Nasze konto:
Pomorski Bank Kredytowy
II Oddział w Szczecinie
numer konta: 368113-25771-136

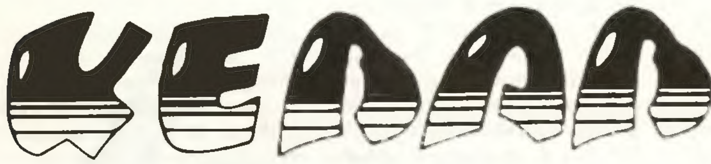
Podajcie dokładny adres, imię i nazwisko zamawiającego oraz numery egzemplarzy, których wpłata dotyczy na odwrocie każdego z odcinków blankietu wpłaty.

I ostatnia prośba - **wszelką korespondencję, wpłaty etc. kierujcie tylko i wyłącznie na adres redakcji.**

REKLAMA

Ogłoszenia drobne od osób indywidualnych (do 10 słów na kuponie wyciętym z III-ciej strony okładki) przyjmujemy bezpłatnie.
Ogłoszenia drobne od osób prawnych oraz zawierające powyżej 10 słów - 1000 zł za słowo.
Ogłoszenia ramkowe (minimalny format 20 cm^2):
1 cm^2 - 4,5 tys. zł, cała strona 2,5 mln. zł,
cała IV strona okładki - 4 mln. zł, 1/2 tej strony - 2,5 mln. zł, dodatkowy kolor - odpowiednio 50 % drożej.

Ogłoszenia płatne prosimy przesyłać listem poleconym.



Nr 2 luty 1993

Własny TURBOLOADER
- dokończenie - na stronie 3

MOTOROLA 68020
- co to takiego? - zobaczycie na str. 9 - 12

POCKET RADIO
- wprowadzenie do tematu - strony 26 - 28

Szybko, szybciej, coraz ...
- czyli zamiast "dopalacza"

CORE WARS 64
- słynna gra do wpisania

Spis treści :

















NOWOŚCI

Jako następca stosunkowo mało znanego programu raytracingowego o nazwie Draw4D wypuszczono nowy pakiet o nazwie Aladdin. Co to jest i z czym to się je poinformujemy wkrótce na naszych łamach.

Od pewnego czasu słychać dużo (i dobrze) o znakomitym pakiecie do Ray-Tracingu o nazwie Real-3D. Oficjalny dystrybutor Real'a zgotował swoim zarejestrowanym użytkownikom miłą niespodziankę. Niespodzianka ta, to seminarium, w którym twórcy programu dzielą się z użytkownikami swoimi doświadczeniami i wiedzą na temat rozmaitych "sztuczek" w pracy z "Real'em". Planowane są również kursy wprowadzające w nowe możliwości przygotowywanej do sprzedaży wersji 2.0.

Również producenci urządzenia o nazwie DCTV (nie CDTV) mają niespodzianki dla swoich klientów. Tym razem chodzi o specjalne udoskonalenie urządzenia, tak aby mogło ono współpracować z magnetowidami typu S-VHS oraz Hi8. Oryginalna DCTV to bardzo popularne rozszerzenie graficzne dla Amigi, które miało jak dotąd niestety możliwość generowania jedynie sygnału typu composite video. Jakość takiego sygnału wystarcza co prawda do celów amatorskich, jednak do poważniejszych zastosowań zupełnie się nie nadaje. Dzięki rozdzieleniu informacji luminancji i chrominancji, uzyskuje się sygnał o znacznie wyższych parametrach jakościowych.

Są już pierwsze "update'y" do znanych programów, umożliwiające współpracę z AA (AGA). Deluxe Paint 4.5 to pierwszy Dpaint obsługujący nowe tryby graficzne. Oprócz tego jest już na rynku nowa ScalaMM o numerze wersji 2.02. Również i ten pakiet jest obecnie w stanie wykorzystać nowe możliwości A1200/4000.

Firma Gold Disk wypuściła pospiesznie nową wersję swojego sztandarowego produktu o nazwie Proffessional Page 4.0. Oficjalnie jest to również update spowodowany chęcią zapewnienia zgodności z AGA. Za kulisami jednak prawdopodobnie kryje się fakt, że wersja 3.0 nie do końca chciała się rozumieć z systemem operacyjnym 3.0...

Jest już od pewnego czasu nowy Diropus. Wersja 4.0 współpracuje bez kłopotów z AGA oraz posiada jeszcze bardziej, w stosunku do poprzedniej, rozbudowany moduł konfiguracyjny. Trudno sobie wyobrazić, co jeszcze mogli rozbudować autorzy tego pakietu,jeżeli chodzi o konfigurację. Już poprzednia wersja pozwalała w tej dziedzinie na niemal wszystko (!), niemniej fakt pozostaje faktem: DirOpus 4.0 może jeszcze więcej.

Kolejny produkt, który współpracuje z AGA, to ImageFX. Nowy pakiet obróbki obrazu reklamowany przez firmę GVP. Tak, ta znana z doskonałych opracowań sprzętowych firma amerykańska próbuje również wywalczyć sobie pozycję na software'owym rynku Amigi.Dobrym początkiem było już wypuszczenie "CineMorph'a", który wchodzi aktualnie (w nowej wersji) w skład pakietu ImageFX. Co prawda wersja 1.01 nie jest jeszcze pozbawiona drobnych błędów, to jednak dzięki dopracowanemu interfacowi użytkownika jest łatwiejsza w obsłudze niż ADPro (ASDG), czy ImageMaster (Black Belt)

Wprowadzenie AGA jednak nie tylko spowodowało falę nowych wersji znanych programów. Okazało się jakby również bodźcem do pracy dla wielu ludzi. Coraz więcej programistów zauważa, jakie potencjalne możliwości daje system operacyjny Amigi. OS2.0 był w tej dziedzinie krokiem milowym. OS3.0 i AGA to konsekwentny rozwój we właściwym kierunku. Owocem tego są coraz to nowe, nieznane dotychczas programy. Wspomniany powyżej Image FX to tylko jeden z przykładów.

U boku bowiem np. DPaint'a wyrastają ciągle nowi konkurenci. Brilliance, Maxon Paint czy Personal Paint to tylko tytuły z jednej dziedziny, które pojawiły się ostatnimi czasy na całostronicowych reklamach w wielu czasopismach.

Podobny ruch występuje we wszystkich dziedzinach zastosowań komputerów, w których Amiga zarezerwowała sobie silną pozycję. Pozostaje tylko zawołać: Oby tak dalej!!!

Pewne ożywienie możemy również zanotować na rynku C-64. Przykładem może tu być fakt, że pojawiają się ciągle nowe opracowania sprzętowe przeznaczone dla C-64 i 128. Twardy dysk dla C-64 to już nie nowość i nie rewelacja. Co prawda 20MB to dla małego "komcia" znacznie więcej niż dla "peceta" 200MB pojemności twardego dysku, ale w wartościach bezwzględnych 20MB pozostaje tylko 20MB.

Jaką natomiast pojemnością dyskietki mogą pochwalić się "profesjonalne maszyny"? 1,44MB i to wszystko! No i ewentualnie po wprowadzeniu dyskietek ED o ośmiokrotnej gęstości zapisu, pojemność ta wzrośnie do 2,8MB. Tymczasem dla C-64 został opracowany przez amerykańską firmę "Creative Micro Designs Inc." (CMD). napęd trzy i pół calowych dyskietek DD/HD/ED.

Pamiętamy jeszcze napęd 1581? Był to jeden z ciekawszych produktów Commodore. Został on zresztą z niezrozumiałych przyczyn (nie pierwszy tego typu krok w historii Commodore) wycofany z rynku europejskiego mimo iż w wielu czasopismach został wybrany produktem roku. Napęd ten pracował korzystając z dyskietek DD - 3,5 cala i pozwalał po zaformatowaniu na zapisanie prawie 800KB danych. Był to doskonały produkt, w szczególności dla miłośników GEOS'a. Co to ma jednak wspólnego z nowym napędem firmy CMD? Otóż firma ta postanowiła zająć miejsce wycofanego drive'u 1581 swoim produktem o nazwie FD-4000.

Napęd ten jest w pełni kompatybilny z Commodore'owskim 1581 w przypadku korzystania z dyskietek DD. Po sformatowaniu dyskietki typu HD napęd jest kompatybilny z... dwoma stacjami typu 1581! a w przypadku (drogiej) dyskietki typu ED z czterema!!!

Jak to możliwe? Otóż dyskietki HD (ED) mogą zostać podzielone na 2 (4) partycje, z których każda będzie traktowana jak niezależna dyskietka typu 1581 (DD). Szybki rachunek (4x800KB) daje nam... 3,2MB na jednej dyskietce!!! Taką pojemnością nie pochwali się żaden pecet.

Można również zaemulować stację 1541 i podzielić jedną dyskietkę ED na... 19 dyskietek typu 1541. Do FD-4000 firma CMD dołączyła wyczerpującą instrukcję oraz zestaw programów kopiujących oraz driver'ów do GEOS'a i CP/M'u.

Wciąż trudno dostępna Amiga 4000 doczekała się również "zubożonej, młodszej siostry". Dla wszystkich mniej zamożnych klientów firma Commodore oferuje A4000/030 25MHz. Wersja z procesorem 68030 będzie zdecydowanie tańsza (o ponad 1000 DM w Niemczech) i zajmie z pewnością miejsce, od początku do końca zbyt drogiej, Amigi 3000 (T)...

Nowe układy graficzne - "AGA - chips" wywołały duże poruszenie w kręgach firm zajmujących się pisaniem oprogramowania. Jak lawina powstają nowe wersje istniejących programów, a także zupełnie nowe produkcje wykorzystujące nowe możliwości wspomnianych układów. Przykładem tych ostatnich jest nowy program graficzny Brilliance firmy Digital Creations.

Ów program został w całości napisany w assemblerze, przez co jak twierdzą autorzy, większość jego funkcji działa szybciej na zwykłej Amidze zaopatrzonej w procesor 68000, niż te same funkcje w konkurencyjnych programach uruchamianych na komputerach zaopatrzonych w "dopalacze" z procesorem 68030. Hasła te wydają się być nieco przesadzone, tym nie mniej Digital Creations ma nadzieję ustanowić nowy standard odniesienia wśród programów graficznych.

A co na to Electronic Arts? Autorzy DPainta nie wydają się być zastraszeni konkurencją i wypuścili "upgrade" do tego najpopularniejszego obecnie programu graficznego. Jak głoszą zapowiedzi nowa wersja obsługuje 256-cio kolorową palettę, oraz ponad 256.000 kolorów w trybie HAM8. Poza tym DPaint-AGA (bo tak właśnie nazywa się nowa wersja) potrafi odczytywać rysunki wykonane w ponad 16-to milonowej palecie, i konwertować je na tryb w którym aktualnie pracujemy, pomimo tego, że oczywiście nie jest w stanie wyświetlić, ani zapisać 24-ro bitowych danych graficznych.

Oczywiście nowa wersja pozbawiona jest kilku drobnych błędów występujących w poprzednim wydaniu. Chociaż DPaint-AGA przeznaczony jest głównie dla użytkowników nowych układów graficznych, montowanych obecnie w A4000 i A1200, posiada on również możliwość pracy w trybach Super-Hires, oraz Productivity.

JAK ZROBIĆ WŁASNY TURBOLOADER czyli

oprogramowanie stacji 1541/71 część 2.

W dzisiejszym odcinku zajmiemy się programowaniem stacji z poziomu asemblera, a także omówimy w skrócie zasadę działania turboloaderów. Programowanie stacji z poziomu języka BASIC, jakkolwiek bardzo łatwe i szybkie jest całkowicie nieprzydatne dla poważniejszych zastosowań.

Prawie wszystkie wartościowe programy na C-64/128 są obecnie pisane w asemblerze. Jest to jedyna droga, aby w dobie komputerów 32-bitowych, taktowanych zegarami szybszymi niż 50 MHz uzyskać na 8-bitowej "zabawce" przyzwoitą prędkość i jakość programu. Pociąga to za sobą konieczność pisania w asemblerze także procedur obsługi stacji. Konstruktorzy przewidzieli, że użytkownikowi może to być potrzebne i tak napisali system operacyjny komputerów C-64/128, że "przeniesienie" programu BASIC-owego do asemblera jest stosunkowo proste. Spróbujmy coś takiego zrobić.

Poprzednim razem napisaliśmy program odczytujący z dysku wybrany sektor i przesyłający go do komputera. Przyjrzyjmy mu się bliżej. Które rozkazy mają jakiś związek ze stacją? Zaraz na początku dwa razy użyty został rozkaz OPEN, raz posłaliśmy do stacji ciąg bajtów poleceniem PRINT, w pętli pobieraliśmy bajty komendą GET, oraz na końcu użyliśmy CLOSE do pozamykania zbiorów. Ale jak wykonać te same czynności w programie maszynowym?

Zajrzyjmy do mapy pamięci komputera C-64, czy C-128 i przejrzyjmy spis procedur KERNAL-a. Czy widzimy jakieś znajome nazwy? Oczywiście, dwie procedury nazywają się OPEN i CLOSE. A co zrobić z PRINT i GET? Do wydrukowania tekstów na ekran często używa się procedury CHROUT, a do pobrania znaku z klawiatury - CHRIN. Okazuje się, że tych samych procedur można używać w odniesieniu do wszystkich urządzeń wejścia-wyjścia, w tym i do najbardziej nas w tej chwili interesującego urządzenia numer 8.

Aby jednak wszystko poprawnie wykonać poznajmy jeszcze takie procedury jak SETNAM, SETLFS, CHKIN, CHKOUT, CLRCHK. Razem z wcześniej przytoczonymi tworzą one blok procedur całkowicie wystarczający do przeniesienia programu z języka BASIC. Omówmy te procedury bardziej szczegółowo. Na początek OPEN.

Procedurę tą można znaleźć pod adresem $FFC0 w tablicy skoków KERNAL-a. Przypomnijmy, że tablica ta stworzona została po to, aby zwiększyć kompatybilność pomiędzy komputerami Commodore. W dużej części tablice te na C-64 i C-128 się pokrywają. Tak się składa, że wszystkie interesujące nas w tej chwili procedury zarówno na C-64, jak i na C-128 mają te same adresy w tablicy skoków. Oznacza to, że programy, które używają tych procedur poprzez tą tablicę będą poprawnie działały na obu komputerach.

Wróćmy jednak do OPEN. Procedura ta służy do otwarcia zbioru logicznego. Polega to na przyporządkowaniu pliku o podanym numerze do interesującego nas urządzenia zewnętrznego. Wywołuje się ją bez żadnych parametrów (po prostu JSR $FFC0), a co się z tym wiąże wszystkie parametry należy ustawić przed jej wywołaniem. Służą do tego dwie kolejne procedury. Pierwsza z nich nazwana została SETNAM i można ją znaleźć w KERNAL-u pod adresem $FFBD. Przed jej wywołaniem do akumulatora wpisać należy długość nazwy pliku, do rejestru .X młodszy bajt, a do .Y starszy bajt adresu nazwy. Przez "nazwę pliku" rozumiem w tym momencie ten sam łańcuch znaków, jaki w BASIC-u można podać jako czwarty parametr polecenia OPEN.

Jeżeli chcemy wykonać OPEN bez tego parametru wywołujemy SETNAM podając zerową długość nazwy. Nie można o tym zapomnieć! Na komputerze C-128 powstaje oprócz tego inny problem z podawaniem adresu nazwy. Komputer ten ma 2 banki po 64 kB RAM każdy. System operacyjny dopuszcza, aby nazwa znajdowała się w dowolnym banku. Jej adres jest jednak podawany w rejestrach .X i .Y, co w sumie daje 16-to bitowy adres, czyli 64 kB (co całkowicie wystarcza w C-64).



Problem ten rozwiązany został w ten sposób, że przed wywołaniem OPEN oprócz wywołania SETNAM ustawić trzeba dodatkowo komórkę $C7, do której wpisać należy numer banku (0-15), w którym znajduje się nasza nazwa. Druga procedura to SETLFS (adres $FFBA w tablicy skoków). Należy do niej przekazać trzy parametry: numer pliku logicznego w akumulatorze, numer urządzenia w rejestrze .X i adres pomocniczy w .Y. Są to te same parametry jakie podaje się po BASIC-owskim OPEN. To tyle na temat otwarcia zbioru.

Sprawa wysłania, czy pobrania bajtu jest stosunkowo prosta. Tak więc, aby posłać jakiś ciąg bajtów (np. rozkaz) do stacji, należy najpierw wywołać procedurę CHKOUT, podając numer pliku logicznego w rejestrze .X jako parametr. Ktoś mógłby zapytać po co to wszystko. Przecież już podawaliśmy tą wartość przed wywołaniem OPEN. Wszystko ma jednak swój cel. Otwarcie zbioru z podanymi parametrami powoduje tylko ustawienie w komputerze wskaźników, że zbiór o tym, a tym numerze ma dotyczyć konkretnego urządzenia, z konkretnym adresem pomocniczym. Możemy więc otwierać więcej zbiorów na raz (maksymalnie 10). Procedura CHKOUT pobiera z tablicy zbiorów dane o tym,

który nas interesuje, po czym ustawia parametry tegoż zbioru (urządzenie, adres pomocniczy) jako aktualne. (Standardowo, aktualne urządzenie wyjściowe to ekran monitora.) Od tej pory każde użycie procedury CHROUT ($FFD2) powoduje wysłanie bajtu do zdefiniowanego przez nas urządzenia wyjściowego.

Kiedy nie chcemy już niczego wysyłać (na przykład wysłaliśmy już cały rozkaz), wywołujemy CLRCHK ($FFCC), co powoduje przywrócenie ekranu jako aktualnego urządzenia wyjściowego, a klawiatury jako aktualnego urządzenia wejściowego. Pobranie bajtu realizuje się analogicznie. Najpierw CHKIN ($FFC6) z numerem zbioru podanym w .X, następnie GETIN ($FFE4), albo CHRIN ($FFCF), aby pobrać bajt.

W przypadku urządzeń podłączonych do szyny szeregowej nie ma znaczenia której procedury używamy. Obie działają identycznie (czyli czytają bajt i zwracają go w akumulatorze). Gdy nie chcemy już pobierać bajtów wywołujemy po prostu CLRCHK. Pozostała nam już tylko formalność, czyli zamknięcie zbioru. Teoretycznie nie jest to niezbędne.

Należy jednak przyjąć zasadę zamykania wszystkich zbiorów, z których nie zamierzamy już korzystać. W przeciwnym wypadku może okazać się, że gdy będziemy chcieli otworzyć jakieś inne zbiory, komputer nie wykona tego, bo zbiór o takim numerze ciągle jest jeszcze otwarty, czy też otwartych jest już 10 zbiorów. Zbiór zamyka się bardzo prosto. Wystarczy załadować do akumulatora jego numer i wywołać CLOSE ($FFC3 w tablicy skoków).

To chyba już wszystko. Nic nie stoi więc na przeszkodzie, aby nasz pierwszy program, który napisaliśmy ostatnim razem napisać w języku maszynowym. Oto on:

```
      LDA #$01
      LDX #$08
      LDY #$0F
      JSR $FFBA
      LDA #$00
      JSR $FFBD
      JSR $FFC0
      LDA #$02
      LDX #$08
      LDY #$05
      JSR $FFBA
      LDA #$01
      LDX #<BUF1
      LDY #>BUF1
      JSR $FFBD
      JSR $FFC0
      LDX #$01
      JSR $FFC9
      LDX #$00
LOOP1 LDA BUF2,X
      JSR $FFD2
      INX
      CPX #$0B
      BNE LOOP1
      JSR $FFCC
      LDX #$02
      JSR $FFC6
      LDX #$00
LOOP2 JSR $FFE4
      STA $0400,X
      INX
      BNE LOOP2
      JSR $FFCC
      LDA #$02
      JSR $FFC3
      LDA #$01
      JMP $FFC3

BUF1  .TEXT "#"
BUF2  .TEXT "U1:5,0,18,0"
```

Program ten nie wymaga chyba żadnego dodatkowego komentarza. Wszystkie procedury dopiero co omówiliśmy. Jako trening proponuję przenieść do asemblera ostatni program z poprzedniego odcinka. Przypomnę, że chodzi o procedurę przenoszącą całą pamięć RAM stacji do RAM komputera. Program taki napisany w języku maszynowym wykonuje się z całkiem przyzwoitą prędkością i z całą pewnością jeszcze nieraz nam się przyda.

Zajmijmy się teraz czymś bliżej związanym z tytułem tego artykułu. Mamy przecież w planie napisać własny turboloader. Czas więc zacząć omawianie części składowych takiego programu. Zacznijmy od wyjaśnienia na jakiej zasadzie właściwie coś takiego działa. O celowości stosowania turboloaderów nie trzeba chyba nikogo przekonywać.

Logiczne jest, że im szybciej da się załadować zbiory z dysku, tym lepiej. Niestety, nie da się przyspieszyć odczytu pisząc program tylko w komputerze. Na szczęście dla nas stacja 1541/71 jest w takim samym stopniu komputerem, co C-64, czy C-128. Ma ona swój własny ROM, RAM, własny procesor (prawie taki sam jak C-64/128) i własne porty wejścia/wyjścia. Istnieją także sposoby, aby przekazać sterowanie do programu zawartego w jej pamięci RAM.

Wynika z tego, że każdy turboloader może (a nawet musi) składać się z dwóch części, tak jakby dwóch oddzielnych programów. Jeden z nich wykonuje się w komputerze, a drugi w stacji. Oczywiście oba te programy wykonywane są jednocześnie. Dokładnie tak, jakby stały obok siebie dwa komputery, tyle, że jeden bez klawiatury i monitora. Pozostaje więc tylko problem, jak sprawić, aby w odpowiedniej chwili komputer i stacja wykonywały odpowiednie programy.

Zacznijmy od początku. Załóżmy, że mamy w pamięci operacyjnej komputera kompletny turboloader. Co musi on zrobić najpierw? Oczywiście przenieść odpowiedni program do RAM-u stacji i tam go uruchomić. Da się to zrobić. Są do tego celu odpowiednie polecenia. Następnie komputer musi przesłać do stacji nazwę zbioru, jaki chcemy załadować. Program w stacji zacznie przeszukiwać katalog dysku, aby sprawdzić, czy taki zbiór w ogóle na tym dysku jest zapisany, oraz aby dowiedzieć się, gdzie się zaczyna. Przez cały ten czas komputer czeka. Następnie stacja może odczytać pierwszy sektor zbioru.

I w tym momencie nadchodzi ważna chwila: skomunikowanie się stacji z komputerem. Stacja daje znać komputerowi, że ma już gotowe dane do przesłania, a następnie czeka, aż komputer odpowie, że też "ma czas" aby te dane przyjąć. Dopiero wtedy może rozpocząć się właściwa transmisja. Po przesłaniu sektora stacja czyta następny, a komputer znowu czeka. I tak aż do końca zbioru.

Kiedy program wykonujący się w RAM stacji wykryje, że przesłany już został ostatni sektor zbioru, informuje o tym komputer (aby ten mógł wyjść z procedury ładującej), po czym przekazuje kontrolę nad stacją do programu w pamięci ROM. Dzięki temu po zakończeniu ładowania stacja może w standardowy sposób przyjmować i wykonywać ko-

lejne rozkazy. Ale jak to wszystko zrealizować? Logiczne jest, że stosując program w BASIC nie mamy "raczej" szans na przyspieszenie ładowania. Poza tym w ROM stacji nie ma interpretera BASIC-a. Nie mamy więc wyboru.

Jedyna droga to język maszynowy. Od czego więc zaczniemy? Od przesłania pewnej ilości bajtów z RAM komputera do RAM stacji. Gdzie jednak "szukać" RAM-u stacji? Wiemy, że stacją zarządza prawie taki sam mikroprocesor, jaki działa w C-64/128. Tak samo jak tam potrafi on zaadresować 64 kB pamięci (16-bitowa szyna adresowa).

W bardzo wielkim skrócie mapa pamięci stacji wygląda następująco:

$0000-$07FF - pamięć RAM (tylko 2 kB!!!)

$1800-$400F - rejestry układów wejścia/wyjścia

$8000-$FFFF - pamięć ROM (stacji 1571) W 1541 ROM zaczyna się od $C000

Pozostałe obszary nie są do niczego podłączone. Praktycznie, własny program umieszczać można tylko w obszarze $300-$7FF. Nie mamy więc zbyt wiele miejsca. Wynika z tego, że część turboloadera wykonująca się w RAM stacji musi być możliwie krótka. Dla nas jednak na razie najistotniejsze jest przesłanie programu do stacji.

Załóżmy, że chcemy przenieść obszar od $2000 do $21FF w komputerze pod adres $300 w stacji. Zacznijmy od napisania takiego programu w BASIC-u, wykorzystując metodę bezpośredniego dostępu:

```
10 OPEN 1,8,15
15 FOR A=0 TO 1
20 OPEN 2,8,5,"#"+STR$(A)
25 PRINT#1,"B-P:5,0"
30 FOR B=0 TO 255
35 PRINT#2,CHR$(PEEK(8192+256*A+B));
40 NEXT
45 CLOSE2
50 NEXT
55 CLOSE1
```

Jako "zadanie domowe" proponuję przeniesienie tego programu do asemblera. Wyniki jego działania łatwo można obejrzeć używając naszej procedury czytającej cały RAM stacji (powinniśmy mieć już napisaną jej wersję języku maszynowym.) Następnym razem poznamy inne stosowane metody przesyłania programu, różne sposoby jego uruchamiania, oraz bliżej przyjrzymy się sprzętowi, czyli dokładniej omówimy mapę pamięci stacji.

Oprócz tego, w najbliższym odcinku znajdą się listingi wszystkich programów, które proponowałem dzisiaj napisać samodzielnie. Pozwoli to na ich porównanie i wyjaśni wątpliwości, jakie być może pojawią się przy samodzielnym kodowaniu...

Krzysztof Matula

Szperam Pascalem dla rozrywki!

Swego czasu w magazynach poświęconych rozrywkom umysłowym często pojawiały się zadania polegające na wyszukaniu słów bądź zdań ukrytych w gąszczu chaotycznie porozmieszczanych liter. "Gąszcz" przybierał zwykle kształt kwadratu czy prostokąta, zdarzały się jednak i bardziej wymyślne konstrukcje. Jedno z takich zadań - wygrzebane z babcinej sterty starych "Rozrywek" - wciągnęło mnie na tyle, iż postanowiłem zainteresować nim moją Amigę. Z dobrym skutkiem.

Tak w skrócie wygląda historia programu "Szperacz", który znajdziecie (wyszperacie?) w dziale "listingi". Jak chyba nietrudno zauważyć został on napisany w Pascalu, a konkretnie w dialekcie *HighSpeed Pascal* firmy *HiSoft*. Potrafi odszukać dowolny ciąg znaków w zadanej macierzy prostokątnej, przy czym ciąg może być napisany wspak, pionowo, ukośnie, itd. - krótko mówiąc na każdy z ośmiu najczęściej spotykanych sposobów. Zastosowany algorytm jest bardzo prosty i nie uwzględnia "odbijania" się słów od brzegów macierzy - pozostawiam to do realizacji Czytelnikom (i bez krzywych żartów proszę - ja też wiem jak to zrobić).

Jako, że listing ten przeznaczony jest dla osób zgłębiających w większym stopniu Pascala niż stare numery "Rozrywki", krótkie objaśnienie procedur składających się na "Szperacza":

- główny blok ma za zadanie pobierać kolejne słowa z tablicy tekstów do znalezienia i przekazywać je do procedury **Szukaj**, po czym wydrukować całą macierz, już ze wskazanymi wyszukanymi wyrazami,

- procedura **Szukaj** przeszukuje wszystkie elementy macierzy pod kątem zgodności z pierwszym znakiem ciągu umieszczonego w zmiennej **Slowo** - i gdy taka zgodność zachodzi przekazuje sterowanie do procedury **Sprawdz**,

- **Sprawdz** to praktycznie jedynie wywołania procedury **Porownaj** dla każdego z ośmiu kierunków,

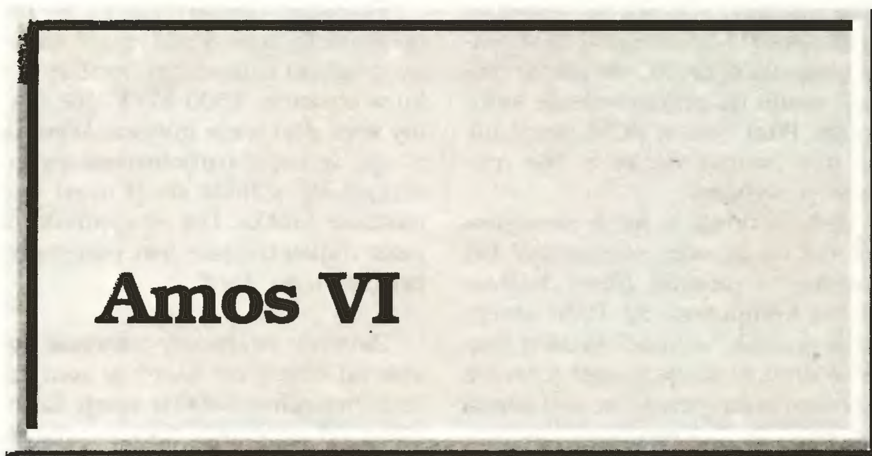
- **Porownaj**, wykorzystując przekazane jej parametry porównuje (no kto by się spodziewał?) łańcuchy znaków leżące w sąsiedztwie badanego

pola ze zmienną **Slowo** - jeśli są identyczne, szukany ciąg pojawia się na ekranie i wołana jest procedura **Zaznacz**...

- ...która korzystając z tych samych parametrów zaznacza ciąg w macierzy zmieniając odpowiadające mu litery na wielkie.

I tyle. Aż żal bierze na myśl, że terminy nadsyłania rozwiązań do owych "Rozrywek" minęły na lata przed pojawieniem się Amigi na rynku.

Miłosław "Thorgal" Smyk



Amos VI

W tym miesiącu zgodnie z zapowiedzią "rozpracujemy" komendy tekstowe. No to do dzieła.

KOMENDY TEKSTOWE

Niektóre instrukcje znamy już z poprzednich numerów, lecz z wymogów formalnych należy umieścić je w odpowiednim dziale.

`Print A$` - drukuje tekst lub wartości zmiennych.

`Centre A$` - j.w, ale centruje tekst.

Komendę Print możemy urozmaicić poprzez zastosowanie funkcji Using. Tak wygląda format tej operacji:

`Print Using` "tekst ze znakami sterującymi";zmienne

Cóż są te znaki sterujące? Otóż i one:

~ - zamiast tych znaków pojawią się w wydruku kolejne litery z pola "zmienne".

- jak wyżej, ale te znaki zostaną zastąpione cyframi. Jeżeli zmienna zostanie podana z częścią ułamkową, to cyfry po przecinku (a raczej kropce), zostaną zignorowane.

+ - przed podaniem liczby, dodany zostaje znak "-" dla liczb ujemnych, lub "+" dla liczb dodatnich.

- - podobnie jak wyżej, ale przed liczbami dodatnimi, pojawi się tylko spacja.

. - zabrania ignorować część ułamkową liczby - można ją wreszcie wyświetlić.

; - zastosowanie jak wyżej, ale nie wyświetla kropki dziesiętnej (w jej miejscu pojawia się spacja).

^ - umożliwia wydruk liczby w postaci wykładniczej.

```
Print Using "Tu beda litery: ~~~~";"Amos"

Print Using "Tu beda cyfry: ######";123456

Print Using "Tu dodamy plusik albo minusik +####";12,-34

Print Using "Tu dodamy tylko minusik -####";12,-34

Print Using "Tu wydrukujemy czesc ulamkowa ###.##";123.45

Print Using "Tu nie bedziemy drukowac kropki ###;##";123.45

Print Using "#.#^^^^";1.2*10^12
```

Skoro są przykłady, to chyba wszystko zrozumiałe...

`Paper N` - ustala kolor N jako kolor tła.

`Pen T` - ustala kolor T jako kolor tekstu.

Rozkazy `Paper` i `Pen` odnoszą się do wszystkich następnych instrukcji `Print` lub `Centre`. W przypadku kiedy chcemy wydrukować tekst różnokolorowy, to nie ma sensu ciągła zmiana Pen czy Paper.

Posługujemy się wtedy funkcjami `Pen$(T)` i `Paper$(N)`. Funkcje te działają podobnie jak ich bliźniaki bez znaku $, z tą tylko różnicą, że umieścić je można w jednej linii z `Print` lub `Centre`, wielokrotnie manipulując kolorami wydruku. na przykład zapis:

```
Print Paper$(0)+Pen$(1)+"a"+Pen$(2)+"b"+Pen$(4)+"c"
```

drukujący "abc" w trzech różnych kolorach, wyręcza nas od pisania sporej procedurki (na przykład siedem linijek to już sporo jak na taką błahostkę), robiącej dokładnie to samo.

O trybie wypisywania tekstu decyduje komenda `Writing W1,W2`, gdzie parametry w1 i w2 oznaczają kolejno:

`w1=0` - standardowy tryb - nowy tekst kasuje stary.

`w1=1` - tryb OR

w1=2 - tryb XOR

w1=3 - ignoruje komendy tekstowe.

w2=0 - tryb standardowy.

w2=1 - wyświetlane jest tylko tło (w miejscu liter zostają "dziury").

w2=2 - wyświetlany jest tylko tekst na tle w kolorze 0 (aktualny kolor ustawiony przez Paper jest ignorowany).

Dodatkowe urozmaicenia wprowadzają rozkazy:

Inverse On - zamienia kolory tła i atramentu (wyłączenie przez Inverse Off).

Shade On - powoduje wyświetlanie "postrzępionego" tekstu (wyłączenie przez Shade Off).

Under On - podkreśla drukowany tekst (wyłączenie przez Under Off).

Wiemy jak wypisać na ekranie tekst, nie wiadomo tylko jeszcze gdzie będzie się on pojawiał. Otóż wszelkie dane tekstowe i liczbowe, wypisywane są począwszy od pozycji kursora.

Po każdej instrukcji Print, kursor przechodzi do pierwszej kolumny następnego wiersza, po Cls ustawia się on w pozycji 0,0, czyli w pierwszej kolumnie pierwszego wiersza (lewy, górny róg).

Aby sterować wydrukiem, przed każdą instrukcją wypisującą coś na ekranie, należy ustawić kursor w odpowiedniej pozycji. Oto przydatne nam komendy:

Locate K,W - ustawia kursor w K-kolumnie (pozycja pozioma) i w W-wierszu (pozycja pionowa).

Cmove X,Y - przesuwa kursor o X pozycji w prawo i Y pozycji w dół. Oczywiście gdy jako parametrów użyjemy liczb ujemnych, to kursor będzie się przesuwał w przeciwnych kierunkach.

Cleft, Cright, Cdown, Cup przesuwają kursor o jedną pozycję odpowiednio w lewo, prawo, w dół i w górę.

W komendach Print, możemy używać funkcji Cleft$, Cright$, Cdown$ i Cup$, na zasadzie podobnej do zastosowania np. Pen$, czyli w jednej linii z Print. Na przykład:

```
Print "abc" + Cdown$
```

Podobną do powyższych rozkazów jest funkcja Tab$, która przesuwa kursor o wiele pozycji w prawo. Rozmiar przesunięcia zmieniamy rozkazem Set Tab N, gdzie N jest tym przesunięciem.



Home - ustawia kursor na współrzędnych 0,0.

Tekst w zadanych współrzędnych pomaga nam wypisać funkcja At(K,W), np:

Print At(1,10);1000 - drukuje w pierwszej kolumnie i w dziesiątym wierszu liczbę tysiąc.

X Curs - funkcja przyjmująca wartość numeru kolumny, w której aktualnie znajduje się kursor.

Y Curs - Podobna funkcja, ale odnosząca się do wiersza.

Memorize X - funkcja zapamiętująca kolumnę.

Memorize Y - jak wyżej, ale zapamiętuje wiersz.

Remember X - przenosi kursor do kolumny zapamiętanej uprzednio przez Memorize X.

Remember Y - zgadnijcie sami...

Cline N - kasuje N znaków na prawo od kursora. W przypadku kiedy parametr N został pominięty, skasowana zostanie cała linia.

Może się zdarzyć, że będziemy chcieli wydrukować coś na ekranie, ale mamy jedynie współrzędne graficzne np: (A,B). Aby dowiedzieć się jakie współrzędne tekstowe powinna posiadać litera, by została umieszczona jak najbliżej naszego punktu graficznego, stosujemy funkcje:

X Text(A) - otrzymujemy z niej kolumnę.

Y Text(B) - otrzymujemy wiersz.

Może równie dobrze zajść potrzeba znalezienia współrzędnych graficznych odpowiadających tekstowym. Stosujemy wtedy funkcje odwrotne:

X Graphic(K) - otrzymujemy współrzędną X.

Y Graphic(W) - a tu Y.

No i w tym oto momencie doszliśmy do samego kursora, jako widocznego "znaczka" na ekranie. Poznęcajmy się trochę nad nim.

Curs Off - chowa kursor. Mimo ,że go nie będzie widać, wszystkie rozkazy z nim związane będą działać prawidłowo.

Curs On - włącza kursor.

Curs Pen N -wybiera kolor kursora. Standardowo jest to migający kolor 3. Sposób migania można oczywiście zaprogramować komendą Flash (patrz: Kebab 11-12/92).

Set Curs A1,A2,A3,A4,A5,A6,A7,A8 - definiuje wygląd kursora.

Ha! Ładnie powiedziane: "definiuje wygląd kursora", ale jakich parametrów użyć? Otóż kursor składa się ośmiu wierszy po osiem pikseli. Podobnie litery.
Oto jak wygląda na przykład litera a:

```
linia 1  %00000000  =0
linia 2  %00000000  =0
linia 3  %00111100  =60
linia 4  %00000110  =6
linia 5  %00011110  =30
linia 6  %01100110  =102
linia 7  %00111011  =59
linia 8  %00000000  =0
```

Teraz to już proste: kolejne parametry odpowiadają za kolejne linie, czyli kursor w kształcie litery a definiujemy następująco:

```
Set Curs 0,0,60,6,30,102,59,0
```

Proste? No to lecimy dalej.

Repeat$(A$,N) - funkcja generująca zmienną tekstową w postaci N powtórzeń zmiennej A$.

Space$(N) - funkcja generująca ciąg złożony z N spacji.

Left$(A$,N) - zmienna tekstowa przyjmująca postać pierwszych N znaków ze zmiennej A$.

Right$(A$,N) - podobnie jak wyżej, ale wybiera ostatnie N znaków.

Mid$(A$,N,M) - wybiera M znaków począwszy od znaku o numerzre N+1.

Powyższych trzech funkcji można używać nie tylko do wybierania ciągów znaków, ale i do wstawiania podanych przez nas tekstów do zmiennych.

String$(A$,N) - tworzy łańcuch tekstowy złożony z N pierwszych liter ze zmiennej A$.

Instr(A$,B$,N) - funkcja przyjmująca wartość położenia ciągu B$ w zmiennej A$. W przypadku kiedy B$ nie zostanie znaleziony w A$, to funkcja przyjmie wartość zero. Parametr N określa od którego miejsca będzie rozpoczęte poszukiwanie (w przypadku jego pominięcia, przeszukana zostanie cała zmienna A$).

Len(A$) - przyjmuje wartość równą długości A$.

Upper$(A$) - przyjmuje postać zmiennej A$, ale złożonej wyłącznie z dużych liter.

Lower$(A$) - jak wyżej, lecz z małych.

Flip$(A$) - zmienna o odwróconej kolejności liter ze zmiennej A$.

Asc(A$) - przyjmuje wartość kodu ASCII pierwszego znaku ze zmiennej A$.

Chr$(N) - funkcja odwrotna do powyższej.

Val(A$) - przekształca zmienną tekstową na liczbową.

Str$(N) - funkcja odwrotna do powyższej.

Zone$(A$,N) - powoduje, że miejsce wyświetlenia zmiennej A$ będzie strefą (o numerze N) rozpoznawalną przez myszkę.

Border$(A$,N) - otacza wydruk zmiennej A$ ramką o numerze N.

Hscroll X - przesuwa tekst w poziomie w zależności od parametru X. Za X podajemy:

1 - przesuwa w lewo linię w której znajduje się kursor.
2 - przesuwa w lewo cały tekst na ekranie.
3 - przesunięcie w prawo linii z kursorem.
4 - przesunięcie w prawo całego tekstu.

Vscroll Y - przesuwa tekst w pionie. Używane parametry X:

1 - przesunięcie w dół tekstu z linii z kursorem i poniższych.

2 - przesunięcie w dół tekstu i linii powyższych. Tekst z linii z kursorem znika.

3 - przesunięcie w górę tekstu z linii z kursorem i powyższych. 4 - przesunięcie w górę tekstu i linii poniższych. Oczywiście tekst z linii w której znajduje się kursor znika.

I to już wszystkie instrukcje tekstowe. Następnym razem powiemy sobie o trochę innym (niby ten sam, a jednak...) Amosie. Mam tu na myśli jego wersję o nazwie Easy Amos.

Zbigniew Piotrowicz.

Jak skorzystać z przerwań graficznych ?

64

W poprzednim numerze KEBABA mój redakcyjny kolega opisał Wam zasadę obsługi przerwań rastrowych lub graficznych, jak kto woli. Do czego to jednak można wykorzystać? Jak to już pewnie wiadomo, stosowanie tego trybu przerwań skutecznie oddaje nam do dyspozycji sterowanie wyświetlaniem komputerowego obrazu. W ten sposób możemy zarządzić np. pokazanie górnej połowy obrazu jako obrazka w trybie wysokiej rozdzielczości a dolnej jako opisu tekstowego na ekranie tekstowym.

Taaak... łatwiej powiedzieć niż samemu zrobić. Aby przełamać opory stawiających pierwsze kroki w assemblerze Czytelników, postanowiłem powolutku przyzwyczajać ich do różnych sposobów wykorzystania przerwań rastrowych. Dzisiaj, na dobry początek, coś mniej widowiskowego. Zacznijmy od sprecyzowania naszych potrzeb: np. w naszym programie napisanym w Basic potrzebowalibyśmy jakiegoś napisu informacyjnego (ot, choćby instrukcji do naszego dzieła). Nie byłby on potrzebny cały czas a tylko w miarę potrzeb, więc najlepiej by było, gdyby można go było pokazywać po naciśnieciu (i przytrzymaniu w tej pozycji) jakiegoś, wybranego klawisza. Do tego celu najlepiej wybrać klawisz, którego wciśnięcie nie powoduje wydrukowania odpowiadającej mu litery czy znaku, wobec czego zdecydowałem się na CTRL.

Aby nasza wyświetlana informacja nie przykryła nam na ekranie innych ważnych danych, zdecydowałem, że najlepszą techniką do demonstracji naszego tekstu powinien być scroller, czyli tzw. "płynący tekst". Aby jeszcze to uczynić wygodniejszym, prezentowana przeze mnie procedura wyświetla płynący tekst w taki sposób, że pojawia się on przy wciśniętym klawiszu CTRL a po jego zwolnieniu rekonstruuje znajdujące się poprzednio w tym miejscu informacje.

W KEBABIE zamieszczono tekst źródłowy INFO-SCROLL'a spod TurboAssemblera wraz z komentarzami. Po niewielkich przeróbkach może być on wpisany pod dowolnym innym assemblerem. Tekst do wyświetlenia należy umieścić pod etykietą TEKST w formie kodów ASCII i zakończyć go kodem zero ($00), co będzie stanowić informacje dla procedury o konieczności zapętlenia tekstu.

Paweł "Polonus" Sołtysiński

MOTOROLA 68020

Najnowszy model Amigi - A1200 został wyposażony w procesor MC68EC020. Jest to pierwszy z serii procesorów 32 bitowych Motoroli, a jego następcy to MC68030, 040 i 060. Procesor MC68EC020 posiada 32 bitową szynę danych, natomiast szyna adresowa ograniczona została do 24 bitów, co daje przestrzeń adresową o wielkości 16 MB (jest to jedyna różnica w stosunku do zwykłego MC68020). Wyposażony jest w 256 bajtów pamięci podręcznej (ang. cache), co przyspiesza znacznie wykonywanie pętli. Do procesora można podłączyć zewnętrzny układ MMU (Memory Management Unit) MC68851 i koprocesor zmiennoprzecinkowy MC68881 lub szybszy MC68882.

MC68020 posiada szereg nowych rejestrów, służących do kontroli pracy procesora. Do zmiany ich zawartości służy uprzywilejowana instrukcja `MOVEC (Move Control Register)`. Także "stara" instrukcja `MOVE SR,<ea>` jest teraz uprzywilejowana i aby z niej skorzystać musi być ustawiony bit S (tryb nadzorcy).

1. ISP (Interrupt Stack Pointer) - wskaźnik stosu przerwania. Dotychczasowy wskaźnik stosu nadzorcy SSP (Supervisor Stack Pointer) został podzielony na dwa rejestry ISP i MSP, przy czym ISP odpowiada rejestrowi SSP w starszych procesorach. Za pomocą odpowiednich bitów w rejestrze statusowym możemy wybrać, z którego wskaźnika chcemy korzystać.

2. MSP (Master Stack Pointer) - wskaźnik stosu zarządcy.

3. VBR (Vector Base Register) - baza wektorów przerwań. Rejestr VBR jest 32 bitowy i umożliwia umieszczenie wektorów przerwań w dowolnym miejscu pamięci, a nie tylko w obszarze $000-$3FF, standardowo jest on wyzerowany.

4. SFC (Source Function Register) - rejestr funkcyjny źródłowy. Rejestry SFC i DFC w połączeniu

z uprzywilejowaną instrukcją MOVES umożliwiają realizację ochrony obszarów pamięci, SFC odczytu, a DFC zapisu.

5. DFC (Destination Function Register) - rejestr funkcyjny docelowy.

6. CACR (Cache Control Register) - rejestr sterujący pamięci podręcznej. Rejestr ten posiada aktywne cztery bity:

bit 0 (E - Enable Cache) - włączanie/wyłączanie pamięci podręcznej,

bit 1 (F - Freeze Cache) - zamrożenie zawartości pamięci podręcznej,

bit 2 (CE - Clear Entry) - kasowanie jednego wpisu w pamięci podręcznej,

bit 3 (C - Clear Cache) - kasowanie całej pamięci podręcznej.

7. CAAR (Cache Address Register) - rejestr adresowy pamięci podręcznej. Rejestr ten jest używany w połączeniu z bitem CE rejestru CACR. Posiada aktywne bity 2-7 (bity 0-1 wyzerowane) umożliwiające wskazanie długiego słowa (wpisu) w pamięci podręcznej.

8. SR (Status Register) - rejestr statusowy. Nie jest to co prawda nowy rejestr, lecz posiada nowe bity:

bit 12 (M - Master/Interrupt State) - wybór wskaźnika stosu MSP/ISP (bit S musi być także ustawiony),

bit 14 (T0 - Trace Enable Extra Flag) - ustawienie tego bitu, przy wyzerowanym T1 włącza nowy tryb śledzenia, który wywołuje przerwanie nie co każdą instrukcję, lecz po rozkazach skoku,

bit 15 (T1 - Trace Enable Main Flag) - w MC68000 oznaczony jako T.

Procesor MC68020 umożliwia operacje na słowie (ang. word) lub długim słowie (ang. long word) znajdującym się pod nieparzystym adresem, trwają one jednak dłużej niż gdy korzysta się z adresów parzystych. Także niektóre rozkazy mogą korzystać z rozmiaru bajtu przy operacjach na rejestrach adresowych (np. CMP2).

Motorola 68020 posiada sześć nowych trybów adresowania. Ich zapis wymaga skorzystania z tzw. nowej składni (ang. new syntax). Różni się ona od starej tym, że wartość przesunięcia jest umieszczana wewnątrz nawiasu, oddzielona od pozostałych składników adresu efektywnego przecinkiem np. `$12,a0` zamiast `$12(a0)`.

1. Tryb adresowania pośredniego rejestrem adresowym z przesunięciem i indeksem (ang. address register indirect with base displacement and index). Składnia asemblera jest następująca:

```
(bd,Am,Dn.L*s)
(bd,Am,Dn.W*s)
(bd,Am,An.L*s)
(bd,Am,An.W*s)
```

gdzie:

n,m - 0..7
bd - 16 lub 32 bitowe przesunięcie
s - współczynnik skalowania równy 1,2,4 lub 8

Tryb ten jest zbliżony do "starego" trybu adresowania pośredniego rejestrem adresowym z indeksem, nowością jest rozmiar przesunięcia wynoszący 16 lub 32 bity (zamiast 8) i możliwość wykorzystania współczynnika skalowania. Przez ten współczynnik będzie pomnożona zawartość rejestru indeksowego przed dodaniem do pozostałych składników adresu efektywnego, gdy go pominiemy domyślnie użyta zostanie wartość 1.

2. Tryb adresowania pośredniego licznikiem programu z przesunięciem i indeksem (ang. program counter with base displacement and index).

```
(bd,PC,Dn.L*s)
(bd,PC,Dn.W*s)
(bd,PC,An.L*s)
(bd,PC,An.W*s)
```

Działanie tego trybu jest identyczne jak poprzedniego, tyle, że zamiast An mamy PC.

3. Tryb adresowania pośredniego pamięcią z postindeksacją (ang. memory indirect postindexed). Jest to najbardziej złożony tryb adresowania. Jego składnia jest następująca:

```
([bd,Am],Dn.L*s,od)
([bd,Am],Dn.W*s,od)
([bd,Am],An.L*s,od)
([bd,Am],An.W*s,od)
```

bd - 16 lub 32 bitowe przesunięcie bazowe (ang. base displacement)
od - 16 lub 32 bitowe przesunięcie zewnętrzne (ang. outer displacement)
s - współczynnik skalowania 1,2,4 lub 8

Sposób obliczania adresu efektywnego w tym trybie jest dość skomplikowany, a mianowicie: do zawartości rejestru bazowego dodawane jest przesunięcie bazowe, spod tego adresu pośredniego w pamięci pobierane jest długie słowo (adres przejściowy), do którego następnie dodawana jest zawartość rejestru indeksowego (znakowo rozszerzona do długiego słowa) pomnożona przez współczynnik skalowania i na koniec dodawane jest przesunięcie zewnętrzne.

4. Tryb adresowania pośredniego pamięcią z preindeksacją (ang. memory indirect preindexed).

```
([bd,Am,Dn.L*s],od)
([bd,Am,Dn.W*s],od)
([bd,Am,An.L*s],od)
([bd,Am,An.W*s],od)
```

Tryb ten jest podobny do trybu postindeksowanego, lecz zawartość rejestru indeksowego jest dodawana do adresu pośredniego, a nie przejściowego, czyli przed pobraniem długiego słowa z pamięci.

5. Tryb adresowania pośredniego pamięcią i licznikiem programu z postindeksacją (ang. PC memory indirect postindexed).

```
([bd,PC],Dn.L*s,od)
([bd,PC],Dn.W*s,od)
([bd,PC],An.L*s,od)
([bd,PC],An.W*s,od)
```

Ten i następny tryb posiadają składnię analogiczną do dwóch poprzednich, lecz zamiast bazowego rejestru adresowego występuje licznik programu PC.

6. Tryb adresowania pośredniego pamięcią i licznikiem programu z preindeksacją (ang. PC memory indirect preindexed).

```
([bd,PC,Dn.L*s],od)
([bd,PC,Dn.W*s],od)
([bd,PC,An.L*s],od)
([bd,PC,An.W*s],od)
```

MC68020 posiada oczywiście także nowe rozkazy. Ważną grupę stanowią operacje mnożenia i dzielenia. Niektóre z nich korzystają z liczb zapisanych na 64 bitach zwanych poczwórnym słowem (ang. quad word). Zapisuje się je korzystając z dwóch rejestrów (niekoniecznie kolejnych) za pomocą składni Dh:Dl, gdzie Dh zawiera 32 wyższe, a Dl 32 niższe bity.

Mnożenie 32 * 32 bity z 32 bitowym wynikiem:

```
MULS.L <ea>,Dn
MULU.L <ea>,Dn
```

Mnożenie 32 * 32 bity z 64 bitowym wynikiem:

```
MULS.L <ea>,Dh:Dl

MULU.L <ea>,Dh:Dl
```

Dzielenie 32 / 32 bity z 32 bitowym wynikiem:

```
DIVS.L <ea>,Dn

DIVU.L <ea>,Dn
```

Dzielenie 64 / 32 bity z 32 bitową całkowitą częścią ilorazu w Dq i 32 bitową resztą w Dr:

```
DIVS.L <ea>,Dr:Dq

DIVU.L <ea>,Dr:Dq
```

Dzielenie 32 / 32 bity z 32 bitową całkowitą częścią ilorazu w Dq i 32 bitową resztą w Dr:

```
DIVSL.L <ea>,Dr:Dq

DIVSL.L <ea>,Dr:Dq
```

Drugą ciekawą grupę rozkazów, szczególnie użyteczną do grafiki stanowią operacje na polach bitowych (ang. bit-field). Pole bitowe stanowi ciąg bitów o długości 1 do 32, jego początek **nie musi się jednak pokrywać z początkiem bajtu.** Ważny jest również fakt, że bity pola są numerowane w odwrotnej kolejności niż "zwykłe" - od strony lewej do prawej (tak jak piksele na ekranie). Pole bitowe jest określane trzema parametrami : adresem bajtu bazowego, szerokością i przesunięciem.

Adres bajtu bazowego może być określany za pomocą wszystkich trybów adresowania z wyłączeniem An, -(An), (An)+ i natychmiastowego, a dla instrukcji BFINS odpadają także wszystkie tryby adresowania licznikiem programu PC. Szerokość pola może być zapisana jako dana natychmiastowa, lub za pomocą rejestru Dn i ma zakres 0..31 (co odpowiada szerokości pola 1..32).

Przesunięcie natomiast ma zakres 0..31 gdy jest zapisane jako dana natychmiastowa, a skorzystanie z rejestru Dn zwiększa ten zakres do 32 bitów ze znakiem co odpowiada liczbom od -2^31 do 2^31-1. Przesunięcie mierzone jest od strony lewej do prawej.

Pole bitowe zapisuje się za pomocą składni :
<ea>{<przes>:<szer>}.

BFTST (Test) - testowanie

Składnia :
```
BFTST <ea>{<przes>:<szer>}
```

Instrukcja ta jest odpowiednikiem BTST, bit Z jest ustawiany, gdy w polu nie ma żadnej jedynki, a w przeciwnym wypadku zerowany.

BFCLR (Test & Clear) - testowanie i zerowanie
Składnia :

```
BFCLR <ea>{<przes>:<szer>}
```

Instrukcja jest odpowiednikiem BCLR - testuje wszystkie bity pola ustawiając odpowiednio bit Z, a następnie zeruje całe pole.

BFSET (Test & Set) - testowanie i ustawianie

Składnia :
```
BFSET <ea>{<przes>:<szer>}
```

Odpowiednik BSET - testuje wszystkie bity pola i następnie je ustawia.

BFCHG (Test & Change) - testowanie i zamiana

Składnia :

```
BFCHG <ea>{<przes>:<szer>}
```

Odpowiednik BCHG - testuje wszystkie bity pola i neguje całe pole.

BFEXTU (Extract Unsigned) - ekstrakcja bez znaku

Składnia :

```
BFEXTU <ea>{<przes>:<szer>},Dn
```

Rozkaz ten kopiuje całe pole bitowe do wskazanego rejestru danych, można go przyrównać do MOVE BF,Dn. Kolejność bitów pola nie jest zmieniana, a całe pole przesuwane jest do bitu 0 rejestru.

BFEXTS (Extract Signed) - ekstrakcja ze znakiem

Składnia :
```
BFEXTS <ea>{<przes>:<szer>},Dn
```

Rozkaz podobny do BFEXTU, lecz bit 0 pola jest kopiowany w górnych bitach rejestru.

BFINS (Insert) - wstawianie

Składnia :
```
BFINS Dn,<ea>{<przes>:<szer>}
```

Instrukcja ta kopiuje zawartość rejestru do określonego pola bitowego, czyli działa jak MOVE Dn,BF.

BFFFO (Find First One) - szukanie pierwszej jedynki

Składnia :
```
BFFFO <ea>{<przes>:<szer>},Dn
```

Instrukcja ta umieszcza w rejes-

trze Dn wartość przesunięcia dla pierwszego napotkanego od lewej strony ustawionego bitu w polu. Gdy nie zostanie napotkana żadna jedynka w Dn znajdzie się wartość <przes+szer>.

CMP2 (Compare Register Bounds) - porównanie zawartości rejestru

Składnia :
`CMP2 <ea>,Dn`
oraz `CMP2 <ea>,An`

Rozmiar : L, W, B

Instrukcja `CMP2` porównuje,czy zawartość rejestru mieści się w zakresie określonym dwiema kolejno zapisanymi pod <ea> granicami (najpierw dolna, potem górna). Ustawia bit Z, gdy zawartość rejestru jest równa jednej z granic i bit C, gdy znajduje się ona poza granicami.

CHK2 (Check Register Bounds) - kontrola zawartości rejestru

Składnia :
`CHK2 <ea>,Dn`
oraz `CHK2 <ea>,An`

Rozmiar : L, W, B

Rozkaz podobny do `CMP2` lecz dodatkowo, gdy zawartość rejestru nie mieści się w granicach inicjuje stan wyjątkowy CHK. Także "stara" instrukcja CHK posiada drugi rozmiar - długie słowo.

EXTB (Extend Sign) - rozszerzenie znakowe

Składnia :
`EXTB Dn`

Rozmiar : L

Rozkaz `EXTB.L` rozszerza znakowo daną wielkości bajtu do rozmiaru długiego słowa. Jest on po prostu połączeniem dwóch instrukcji : `EXTB.W Dn i EXTB.L Dn.`

LINK (Link & Allocate) - łączenie i alokacja

Składnia :
`LINK An,#<przesunięcie>`

Instrukcja `LINK` procesora MC68020 korzysta teraz także z 32 bitowego przesunięcia oprócz starego - 16 bitowego.

RTD (Return and Deallocate Parameters) - powrót i dealokacja parametrów

Składania :
`RTD #<przesunięcie>`

Rozkaz ten powoduje wyjście z podprogramu jednocześnie zwiększając wskaźnik stosu o 16 bitowe przesunięcie. Instrukcja RTD jest użyteczna wtedy, gdy parametry dla podprogramu przekazujemy z użyciem stosu, zastępuje ona wtedy `ADD.L #<przesunięcie>, SP i RTS.`

Bcc (Branch Conditionally) - skok warunkowy

Składnia :
`Bcc <etykieta>`

Rozmiar : L, W, B

Nowością jest przesunięcie wielkości długiego słowa oprócz bajtu i słowa. Podając rozmiar przesunięcia warto korzystać z oznaczeń .L, .W i .B zamiast .L i .S, gdyż są one mylące.

BSR (Branch to Subroutine) - skok do podprogramu

Składnia :
`BSR <etykieta>`

Rozmiar : L, W, B

Uwagi jak wyżej.

Na koniec mała uwaga dla wszystkich, którzy chcieliby programować z wykorzystaniem instrukcji procesora MC68020 - należy ich używać w uzasadnionych przypadkach, gdy są one niezbędne dla szybkiej pracy programu (pamiętajmy o "upośledzonych" posiadaczach MC68000). Program powinien sprawdzać typ zainstalowanego procesora i ewentualnie informować użytkownika o niemożności uruchomienia, a nie bezczelnie się zawieszać. Do asemblacji należy używać odpowiedniego asemblera, który jest w stanie rozpoznać nowe komendy np. Devpac'a.

Konrad Dubiel

SAINT GROUP

Obok licznych, silnych i powszechnie znanych grup, istnieją na polskiej scenie inne, trochę schowane w cieniu swych potężnych konkurentów. Trudno jest im przebić się na "wyżyny", przede wszystkim ze względu na pewien schemat myślowy: jeżeli grupa *Anarchy* wydała nowe demo, to każdy chciałby je zobaczyć, natomiast w przypadku gdy grupa o mało znanej nazwie ma zamiar się pochwalić swą produkcją, ludzie podchodzą do niej z dużym sceptycyzmem.

Oto właśnie przykład jak dużą rolę odgrywa ustalona pozycja na scenie, zdobywana zresztą dużym nakładem pracy i nawet w ciągu kilku lat.

My, jako redakcja Kebaba, obiecaliśmy zarówno omawiać dorobek zasłużonych grup, jak i promować "nowe twarze" sceny. W tym miesiącu: Święta Grupa z Gryfic.

Saint Group jest stosunkowo nową grupą, założoną w czasie ostatnich wakacji. Obecnie należą do niej: **Judas** (prezes i koder), **Harry** (koder), **Piter** (tłumacz i tekściarz), **Stan** (muzyk) i **Bilbo** (koder).

Tym co wyróżnia ich spośród innych teamów, jest sposób w jaki pracują. Główny programista, czyli Harry, widzi swoją Amigę co najwyżej raz w tygodniu: coś tam zakoduje i zaraz musi wracać z Gryfic do



Szczecina na studia. Bilbo jest gdzieś we Wrocławiu, grafika również ciężko uświadczyć. Mimo przeciwnych prądów, grupa rozpoczęła działalność z rozmachem, wydając pierwszy numer *Imazine'a.*

Imazine to niecodzienny magazyn dyskowy. Otóż użytkownik dostaje do swej dyspozycji wszystkie udogodnienia jakie daje nam system Amigi. Tak, tak, Imazine jest kodowany, jak to się zwykło mówić "pod systemem", dlatego też właśnie mamy pełen multitasking, możliwość definiowania własnych kolorów, posługiwania się pull-down menu. Nic nie stoi również na przeszkodzie, aby rezygnując ze standardowo dołączonych do niego modułów, odtworzać sobie (na przykład Intui Trackerem) podczas czytania inną, dowolną muzyczkę. Również ja pisząc ten tekst, równocześnie przeglądam sobie artykuły. Multitasking to nie tylko wygoda, ale i także ważny element praktyczny.

W menu głownym dostępna jest opcja "Ekranik" blokująca task z magazynem. Wtedy Imazine nie będzie sie odróżniał od innych, podobnych mu programów. Jedyną według mnie niedogodnością jest "wybieraczka" artykułów. Ma ona postać przewijających się tytułów, a wybranie jednego z nich następuje po naciśnięciu lewego przycisku myszy. Otóż problem w tym, że tytuły reagują nawet na drobne ruchy myszki, co wywołuje nieprzyjemne wrażenie drgania. Ponarzekałem? Ponarzekałem, ale to chyba ostatnia rzecz dająca mi tę możliwość.

Oprawie graficznej również niczego nie można zarzucić. Nie jest ona może zbytnio bogata, ale za to estetyczna i miła dla oka. 2 moduły dołączone do pierwszego numeru, nie są skomponowane przez redakcyjnego muzyka, lecz przez Grubiego z zachodniej grupy Brainstorm. Trzeci jest już dziełem rodzimym, całkiem pasującym do wymogów stawianych modułom umieszczanym jako podkład do "magów" - nie usypia, ale i przy tym nie męczy hałaśliwością i dynamiką.

Tyle o stronie technicznej. W magazynach dyskowych liczy się przede wszystkim zawartość. Cóż oferuje nam Imazine? Rozpiętość tematyczna jest bardzo duża: począwszy od opisów gier na charakterystyce przerwań Amigi skończywszy. Nie zabrakło też humoru, wieści ze świata, cheatów i innych pozycji tematycznych, spotykanych we wszystkich disk-magach.

Choć trochę wody w niektórych miejscach się polało, to za to ciekawą nowością jest udostępnienie swych łamów dla innych komputerów: wiele jest ciekawych informacji na temat MacIntoshów, miły artykuł posiadacza Atari ST (dla podkreślenia tego, że podobał mi się ten tekst, napisałem Atari dużą literą), mniej interesująca dyskusja z pecetologiem. Zresztą miarodajną informacją bedzie to, że łączna objętość tekstów to prawie 247 kilobajtów. Ocenę czy to dużo czy mało zostawiam Czytelnikom.

Jak na pierwszy numer, całość napawa optymizmem, pozwalającym spokojnie czekać na kolejne numery. Jeżeli plany redakcji co do regularności wydań się powiodą, poprawiona zostanie opcja wyboru artykułu, oraz zjawią się nowi, wartościowi "tekściarze", to kto wie czy w przyszłości Imazine nie stanie się poważną konkurencją dla takich potentatów jak ZigZag, Fat Agnus, czy Next Life.

Podsumowując pierwszą produkcję Saint Group, można powiedzieć o udanym debiucie. Czekamy na dalsze efekty ich pracy; być może będzie to demo, użytek, a na pewno już kolejne numery Imazine'a. Główną przeszkodą stojącą na drodze do większego poświęcenia czasu Amidze jest oczywiscie szkoła, na którą to użala sie głównie Harry. Jeżeli ktoś chce pomóc przy redagowaniu pisma może pisać na adres redaktora naczelnego Imazine'a:

Krzysztof Więckowicz

ul. Nadrzeczna 3b/7

72-300 Gryfice

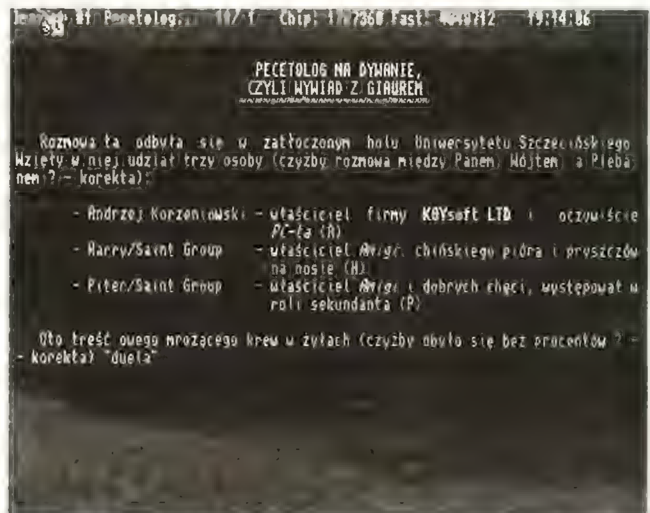
tel.(0-931) 3069

Z Saint Group, a konkretnie z Harrym, można się również skontaktować przez sieć Bitnet:

BK304@PLSZUS11

Zbigniew Piotrowicz.

P.S. W Imazinie spróbujcie wgrać np. inną muzyczkę wówczas, gdy w stacji nie ma dysku z magazynem. I co sądzicie o tekście w requestrze? Szóstka dla kodera za pomysł!



Szybko, szybciej

coraz szybciej...

Język maszynowy jest bez wątpienia niedościgniony w dziedzinie szybkości wykonywania programu. Wielu z was, znudzonych wielogodzinnym oczekiwaniem na wynik pracy programu w BASIC'u próbowało swoich sił w assemblerze. Niektórzy też próbowali pisać różne programy korzystające z przerwań rastra.

I tu pojawiły się pierwsze "schody". Zdawało by się niedościgniony "maszyniak" okazywał się czasami zbyt wolny!!! Nasze intro, demo czy inny program nie mógł zawierać tego wszystkiego co sobie zaplanowaliśmy z powodu braku tzw. czasu rastra. Mimo, że w niektórych demach koderzy zdają sie "rozciągać" go jakby był z gumy, to jednak w rzeczywistości cały czas mamy ograniczoną liczbę cykli do dyspozycji.

Jak więc poradzić sobie gdy nagle okaże się, że w naszym programie mamy zbyt mało czasu rastra i np. nie możemy odgrywać muzyczki? Postaram się udzielić wam kilku rad, które na pewno pomogą wam w zmaganiach z "ramką":

- Program krótki (pamięcio-oszczędny) prawie nigdy nie jest programem szybkim. Dążenie do otrzymania jak najkrótszego kodu wynikowego z reguły katastrofalnie odbija się na szybkości jego działania.

- W naszych programach starajmy się jak najrzadziej używać pętli, gdyż to one są jednym z największych "złodziei" czasu. Jeżeli już z nich korzystamy, to starajmy się albo "upchnąć" w nie jak najwięcej różnych procedurek, które można przystosować do pracy w danej pętli (chodzi głównie o zawartość rejestru indeksującego) albo też starajmy się zmodyfikować kod wewnątrz pętli, tak aby można było ograniczyć ilość przebiegów pętli. Dobrym przykładem może być tu programik wypełniający kawałek ekranu spacjami:

```
        LDA #$20
        LDX #$00
label   STA $0400,x
        DEX
        BNE label
```

Wykonanie tego programu pochłania 2307 cykli a analogiczny program napisany bardziej "pamięciożernie":

```
        LDA #$20
        LDX #$08
label   STA $0400,x
        STA $0408,x
        STA $0410,x
        ...........
        DEX
        BNE label
```

pochłania ich tylko 1067!!!

- Ekstremalnym przykładem przyśpieszania są programy w ogóle nie używające pętli.Są one naprawdę bardzo szybkie, ale niestety pochłaniają straszne ilości pamięci. Należy jednak pamiętać, że czasami (jak w programie 2) pętla pochłania tak znikome ilości cykli, że nie opłaca się jej usuwać.

- Jeżeli nasz program powtarza jakąś złożoną procedurę kilkanaście (-dziesiąt!) razy w czasie przerwania, to powinniśmy poświęcić dużo czasu na maksymalne jej przyśpiesznie. Jeżeli uważamy, że nasz algorytm jest najszybszym z możliwych (co, szczególnie u początkujących, rzadko bywa prawdą!) to spróbujmy napisać go na różne sposoby, za każdym razem obliczając ile cykli pochłania nasza procedura.

Bardzo pomocna jest tu lista rozkazów 6502 wraz z czasami ich trwania. Jeżeli uda nam się zaoszczędzić tylko 5 cykli na procedurze, która jest wykonywana np. 200 razy w ciągu przerwana to skrócimy całość o 1000 cykli czyli ok. 15 linijek rastra!

- Często spotykanym sposobem przyśpieszania programów maszynowych jest tzw. samomodyfikacja. Najłatwiej wyjaśnić to na przykładzie: Jeżeli w każdym przerwaniu musimy powtarzać jakąś pętlę w której raz na jakiś czas zmienia się pewien adres to zamiast stosować któryś z trybów adresowania strony zerowej warto zastosować tryb absolutny indeksowany i modyfikować sam argument danego rozkazu. Przykład programu:

```
        LDY #$00
label   LDA ($a0),y
        STA $0400,y
        INY
        BNE label
```

pochłaniający 3585 cykli warto zastąpić programem:

```
        ldy #$00
label   lda $1000,y
        sta $0400,y
        iny
        bne label
```

w którym będziemy co jakiś czas modyfikować argument rozkazu LDA, a który pochłania o 256 cykli mniej!

- Jednym z najbardziej rozpowszechnionych wśród programistów (a wcale jakby nie dostrzeganych w "poważnych" publikacjach) sposobów przyśpieszania programów jest tablicowanie różnych obliczeń i adresów. Celowość stosowania tablic bardzo łatwo można zrozumieć na przykładzie procedury stawiania punktu w trybie graficznym.

"Wzrocowa" procedura, która liczyła wszystko wewnątrz programu zajmuje ok.3 linijek rastra, a procedura korzystająca z zaledwie z 768 bajtów tablic robi to samo w ok.1/2

linijki!!! Tablicować możemy praktycznie wszystko. Jeżeli np. w prograie jeden z parametrów wejściowych jest mnożny przez 32 to zamiast pracochłonnie mnożyć go przez 2 (10 cykli) wystarczy umieścić go w rejestrze indeksującym i pobrać wartość danej liczby razy 32 z odpowiedniej 256 bajtowej tablicy (6 cykli!).

- Unikajmy w naszych programach stosowania podprogramów wywoływanych przez JSR a już zupełnie odradzam stosowanie JSR wewnątrz pętli wykonywanej wiele razy. Np. wykonanie pętli 256 razy powoduje, że program wydłuża się nam na skutek używania pary JSR+RTS o 3072 cykle!!!

- Instrukcja JMP też nie powinna zbyt często znajdować się w naszym programie. Ogólnie mówiąc najlepiej pisać programy "ciurkiem" bez żadnych podprogramów itp. Nie jest to może zbyt poprawne z punktu widzenia estetyki programu, ale na pewno skuteczne.

- Wracając do pętli. Jeżeli już koniecznie musimy jej użyć to starajmy się tworzyć pętle zliczające od zadanej wartości do zera a nie odwrotnie. Pozwoli to na pominięcie rozkazy CMP a co za tym idzie skróci każdy przebieg o co najmniej 2 cykle!

- Ciekawym sposobem na przyśpieszenie programu jest umieszczenie najczęściej używanej procedury na stronie zerowej, co przy jednoczesnym użyciu szybkich trybów adresowania strony zerowej i samomodyfikacji pozwala skrócić ją nawet o kilkanaście cykli!

- Strona zerowa jest bardzo dobrym miejscem do przechowywania różnych liczników i zmiennych ze względu na szybkość rozkazów z nią związanych.

Gdy to wszystko zawiedzie zawsze pozostaje nam gruntowne przemyślenie i modyfikacja algorytmu, co prawie zawsze daje pożądany efekt.



Wszystkie przykłady przeznaczone są dla Commodore 64 ale będą zrozumiałe dla każdego, kto zna assembler 6502. Wszelkie uwagi ogólne są prawdziwe dla wszystkich typów procesorów.

Na koniec życzę jak najszybszych programów i "gumowego" czasu rastra!



Krzysztof "BRUSH" Dąbrowski

Nowe możliwości

O systemie 2.0 napisano już wiele. Wszystkie publikacje dotyczyły najczęściej zmian wprowadzonych do Workbencha. Nikt dotychczas nie zajął się konsolą, a przecież większość ludzi w kraju, nie mając do dyspozycji twardego dysku, właśnie w ten sposób konwersuje z Amigą. Niniejszym chciałbym przedstawić kilka sposobów mogących ułatwić codzienną pracę z systemem 2.0.

1. Jeżeli nie lubisz komendy ENDCLI, lub ENDSHELL - już więcej nie będziesz musiał jej używać. Alternatywnym rozwiązaniem jest kombinacja klawiszy CTRL oraz znaku \.

2. System 1.3 skazywał nas na jednolity wygląd okienka CLI. Chociaż pojawiły się programiki i bootblocki zmieniające dane w strukturze okna przed jego otwarciem, tak aby wyeliminować niektóre jego elementy, (na przykład gadżet do zmnieniania wielkości) to często nie pracowały one zgodnie z intencjami autora. Użytkownicy systemu 2.0 nie muszą uciekać się już do takich partyzanckich metod. Przy otwieraniu nowego okna można bowiem podać szereg znaczników, których wynik działania zadowolić powinien nawet osobników o najbardziej wysublimowanych gustach. Należą do nich:

CLOSE - dodaje gadżet zamknięcia okna.

NODRAG - eliminuje gadżet przesuwania okna.

NOSIZE - eliminuje gadżet zmieniania wielkości, powiększania i zamykania.

NOBORDER - otwiera okno bez ramek.

BACKDROP - otwiera okno na samym spodzie stosu.

A oto mały przykład:

```
NEWCLI con:0/10/640/150/Kebab/NOBORDER/NOSIZE/CLOSE
```

Dla przypomnienia dodam jeszcze, że pierwsze dwie liczby (0 i 10) są współrzędnymi okna, następne to szerokość i wysokość, jego tytuł a dalej to już wiadomo.

3. Jeżeli wynikiem jakiejś komendy (np dir, type) jest zbyt długi tekst, przelatujący z prędkością większą od Twoich możliwości czytania, to możesz go zatrzymać kombinacją klawiszy CTRL oraz S. Wznowienie wypisywania następuje po sekwencji CTRL Q.

4. W systemie 1.3 nigdy nie było problemem wyświetlenie nazw plików i katalogów z rozszerzeniem na

przykład ".s". Wystarczyło wpisać:

```
dir #?.s
```

Schody zaczynały się w momencie, gdy chcieliśmy wylistować wszystko to, co NIE miało rozszerzenia ".s". System 2.0 rozwiązuje również ten problem wprowadzając operator przeczenia "~". Proszę spróbować:

```
dir ~(#?.s)
```

5. Zupełnie kłopotliwym mogło być kiedyś wylistowanie tych plików z danego katalogu, które zaczynają się na którąś z liter z przedziału na przykład od b do f. Będąc w posiadaniu 2.0 możemy wpisać:

```
dir df0:c/[b-f]#?
```

Wygląda dziwacznie, ale działa.

6. Ponad trzydzieści komend z katalogu C: powędrowało do ROMu. Ich pełną listę możemy otrzymać wydając komendę:

```
resident
```

Owe komendy możemy jednak uczynić niewidocznymi dla systemu (z pamięci ROM raczej trudno byłoby je usunąć - być może jednak profesjonaliści zaproponują jakąś metodę). Uzyskujemy to poprzez wydanie komendy resident z opcją REMOVE:

```
resident <nazwa komendy rezydentnej> REMOVE
```

Jeżeli zaś komendę z ROMu chcemy zamienić komendą dyskową o tej samej nazwie to używamy opcji REPLACE.

7. Jak zapewne zauważyliście wśród ROMowskich komend znajduje się również znana CD. Użytkownicy nowego systemu nie będą musieli jej już więcej używać, gdyż w celu zmiany aktualnego katalogu wystarczy podać jego nazwę. Na przykład zamiast pisać:

```
cd df0:prefs
```

wystarczy:

```
df0:prefs
```

8. Komenda avail otrzymała nowy znacznik, mianowicie FLUSH.

Wykonanie:

```
avail FLUSH
```

spowoduje wyrzucenie z pamięci wszystkie pozostające tam, a już nie używane fonty, biblioteki, urządzenia.

9. Komenda search uzyskała nową opcję FILE, pozwalającą na wyszukanie z danego katalogu plików określonych wzorcem.

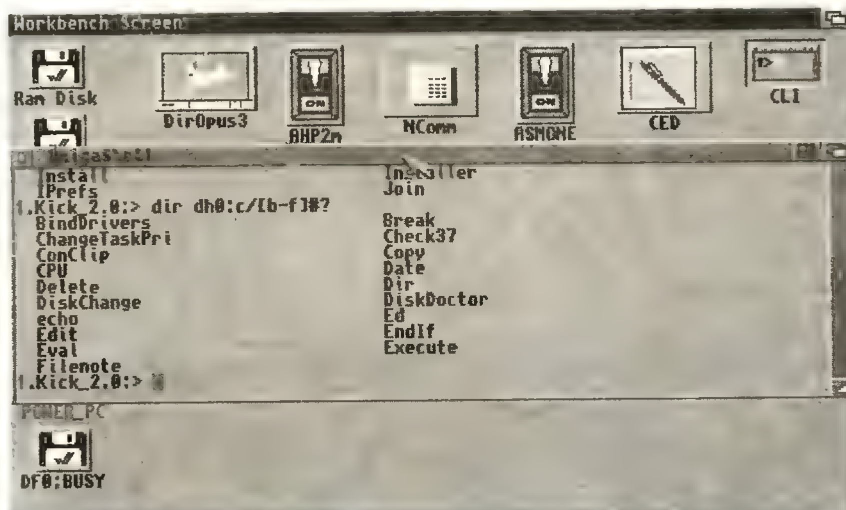
Na przykład:

```
search df0: #?.info FILE ALL
```

Wypisze wszystkie ścieżki dostępu do plików z rozszerzeniem ".info", czyli ikonek.

10. Bardzo interesujące zmiany dotyczą komendy assign. Obecnie mamy możliwość "przyasygnowania", czyli podłączenia danego katalogu logicznego do kilku katalogów fizycznych. Uzyskujemy to wykorzystując na przykład opcję ADD:

```
assign FONTS: df0:MyFonts ram:fonts ADD
```



Powyższy przykład dodaje dwie ścieżki do katalogu logicznego FONTS: Jak zatem zachowa się w takim wypadku komenda:

```
dir FONTS:
```

Otóż wyświetli ona zawartość PIERWSZEGO przyasygnowanego katalogu. W przytoczonym wyżej przykładzie ze względu na opcję ADD nie możemy stwierdzić, który był przyasygnowany jako pierwszy. Możemy to jednak stwierdzić w poniższym przykładzie:

```
assign FONTS: df0:MyFonts ram:fonts
```

i z całą pewnością powiedzieć, że katalogiem tym będzie df0:MyFonts. A oto rozwiązanie problemów osób posiadających jako pamięć zewnętrzną jedynie stacje dysków. Koniec z wachlowaniem podczas uruchamiania programów! Rozwiązeniem jest... opcja PATH. Proszę spróbować:

```
assign LIBS: df0:libs PATH
```

Od teraz wszystkie odwołania do katalogu logicznego LIBS: będą dotyczyły fizycznego katalogu "libs", znajdującego się na dyskietce aktualnie włożonej do stacji "df0:".

na podstawie "Amiga World" opracował:
Krzysztof Kobus

Zamiast mapy pamięci

Pamiętacie te czasy, kiedy do Polski Ludowej zawitały tajemnicze urządzenia elektroniczne sprowadzone przez przedsiębiorczych rodaków ze zgniłego Zachodu? Szokujące możliwości sztandarowego produktu brytyjskiej firmy Sinclair podbiły serca większości Polaków. To było coś niesamowitego! Każdy mógł posiąść tajemną wiedzę programowania w Basicu i już za momencik obejrzeć efekty na ekranie telewizora Rubin.

Nikt się wtedy nie przejmował takimi szczegółami jak rozdzielczość, ilość dostępnych kolorów jednocześnie na ekranie, wielkością palety czy chociażby prędkością komputera. Gdy stało się to ważne (jakieś 2 lata później), społeczeństwo zaczęło rozglądać się za czymś o znacznie większych możliwościach. Jedni (ci smutniejsi o twarzy niebieskiej) zauważyli cud techniki: IBM PC-AT ze wspaniałą kartą CGA (ew. HERCULES), ci drudzy (weseli i bardziej rozwinięci psychicznie) zwrócili się ku komputerom. Mam na myśli oczywiście Amigę 1000, która to w 1985 stała się przebojem wśród tych, którzy mieli sprzęt ośmiobitowy.

Mówiło się o wspaniałej grafice i cudownych wręcz możliwościach jej przetwarzania. Mówiło się i co z tego wyszło? Otóż nic. Po paru latach zaczęły pojawiać się coraz lepsze karty graficzne dla smutnych komputerów, a Amiga stała jakby w miejscu. Niektórzy może zakrzykną w tym momencie wielkim głosem, że na zachodzie ukazywały się tak samo i karty dla Amigi.

Zgoda, tylko, że w Polsce nikt o nich nie słyszał (nawet teraz kiedy znamy już takie nazwy jak Video Toaster, Impact Vision 24 czy Opal Vision to znamy też ich ceny... W półprofesjonalnych (w pełni profesjonalny jest tylko jeden magazyn) magazynach komputerowych typu Bajtek pojawiały się krótkie wzmianki o SVGA z wielkimi nagłówkami 'SVGA czyli CIAO AMIGO'.

No cóż, trzeba przyznać rację, bo rzeczywiście 640x512 w 16 kolorach to nie był szczyt osiągnięć konstruktorów chipów graficznych. Ale jak mawiają starzy czarownicy z plemienia Apaczów, śmiej się, tylko uważaj, bo możesz być ostatni... Nie, to było chyba: śmieje się ten kto jest ostatni? Nie, też nie tak. A może: ten kto śmieje się teraz jest wesoły, a ten co śmiał się wcześniej jest smutny!!! W każdym razie wiadomo o co chodzi. AA-Chipset, prawda?

Układ ten pojawił się początkowo tylko w Amidze 4000. 8 bitplanów w każdej rozdzielczości to było wreszcie coś. Tylko ta cena ... I tu Commodore zdecydował się wreszcie na jeden z niewielu rozsądnych kroków w swojej długiej i burzliwej historii: AA pojawił się także w nowej A1200.

Nie byłoby w tym żadnej rewelacji, gdyby nie cena: 399 funtów albo 899 DM. Za prawdziwy, 32-bitowy system, 2Mb pamięci i doskonałą grafikę to naprawdę niewiele. Tu jednak Commodore zakończył swą passę dobrych kroków i zapowiedział, że nie dostarczy dokumentacji do nowego Chipseta. Dlaczego? Bo i po co, brzmiała odpowiedź.

Przecież wszystko obsługuje KickStart 3.0, no nie? Oczywiście dla programistów "użytkowych" wiedza sprzętowa jest całkowicie zbędna i przez nich samych określana jako "brudna" i "zbrukana stygmatem lamerstwa". Weźmy jednak takich demo-kletów czy piszących tak wspaniałe gry jak Jim Power, Turrican czy Shadow of the Beast III. Oni wręcz przeciwnie, znają na pamięć znaczenie większości sprzętowych rejestrów, a system pobieżnie (to już zależy od indywidualnego podejścia danego kodera do sprawy). I cóż oni, biedni, maluczcy, odtrąceni przez Guru mają począć? Położyć uszy po sobie i zacząć wyrabiać wykałaczki? Nigdy!!! Trzeba walczyć do ostatniego RTS'a. Wynik zmagań z Chipsetem mam właśnie zamiar przedstawić.

Wszystkie przedstawione tu informacje są uzyskane drogą testów praktycznych, a nie uzyskane z jakiegoś źródła z jednego powodu. Z powodu braku takiego źródła. Oczywiście wszystkie dane i znaczenie bitów są prawdziwe, tylko nie udokumentowane. Czyli, jak ktoś nie lubi eksperymentów, to może nie czytać i czekać na łaskę mistrzów marketingu z Commodore'a.

Na pierwszy ogień poszła oczywiście copperlista jako coś ogólnie znanego. Wiemy bowiem, że musiały się gdzieś pojawić rejestry 256 kolorów, jako, że normalnie były na to przeznaczone 32 słowa (od $DFF180 do $DFF1BE).

Uruchamiamy dowolny monitor kodu, znajdujemy systemowe copperlisty i czego się dowiadujemy? Otóż nie ma dodatkowych rejestrów kolorów! Zastosowano stare, rozwiązanie zwane bankowaniem rejestrów. Mamy cały czas do dyspozycji 32 rejestry koloru, tylko, że ustawiając odpowiedni rejestr kontrolny (w tym przypadku banków takich jest 8, bo 8*32=256) kolory te są modyfikowane w określonym przez nas banku.

Przykładowo, chcemy ustawić rejestr koloru numer 137. Dzielimy więc 137 przez 32, 137=4*32+9. Wynik dzielenia (4) jest numerem banku, a reszta numerem koloru (COLOR9=$DFF192). Rejestr, który jest odpowiedzialny za bank zwie się swojsko `BPLCON3 ($DFF106)`.

BPLCON3:

bity 15 - 13 Numer banku

bity 12 - 10 Nieznane, nie używane...

bit 9 Bit kontrolny

Ciekawe po co jest ten bit kontrolny. Chipset posiada bowiem coś takiego jak 24 bitową paletę kolorów. Tak więc do zapamiętania 24 bitów potrzebne są nam 2 słowa, a nie jedno tak jak to było w starej Amidze (pamiętamy: paleta 4096 kolorów). Kolor zbudowany jest z komponentów czerwonego, zielonego i niebieskiego, a każdy może zmieniać się w zakresie od 0 do 255. Tak więc paleta obejmuje 16777216 kolorów. Przedstawiając liczbę określającą kolor w zapisie szestnastkowym: $RrGgBb otrzymujemy jednocześnie przejrzystą informację o wartości każdego komponentu. Na przykład: kolor zielony to liczba $00FF00, a żółty to $FFFF00 itp. Bit 9 w BPLCON3 określa nam która cyfra komponentu jest ustawiana w banku. Jeśli bit ów jest wyzerowany, to ustawiamy cyfry bardziej znaczące, jeżeli natomiast ustawiony to mniej znaczące. Stanie się to chyba bardziej przejrzyste po przeanalizowaniu poniższego przykładu.

Chcemy wpisać do rejestru koloru tła, kolor o składowych R=$F1, G=$45, B=$B6. Łącząc te liczby otrzymamy $F145B6. Wyodrębniamy cyfry bardziej znaczące i tworzymy z nich liczbę: $F4B oraz mniej znaczące: $156. Teraz pozostaje już tylko wpisać do copperlisty :

```
DC.W    $106,$C00
DC.W    $180,$F4B
DC.W    $106,$E00
DC.W    $180,$156
```

W ten sposób postępujemy ze wszystkimi kolorami z palety - warto napisać do tego zadania krótką procedurkę.

* * *

Ponieważ pojawiły się dwa kolejne bitplane'y, musiały się też pojawić wskaźniki (pointery) do nich. Są one umieszczone tam gdzie być powinny, czyli bezpośrednio po BPL6PTL:

```
BPL7PTH =$DFF0F8
BPL7PTL =$DFF0FA
BPL8PTH =$DFF0FC
BPL8PTL =$DFF0FE
```

Pozostaje już tylko dowiedzieć się gdzie włącza się 8 bitplanów. Standardowo ilość bitplanów podawało się w BPLCON0 ($DFF100) z odpowiednim przesunięciem. Przykładowo chcemy włączyć 5 bitplanów, wpisujemy więc w copperliście:

```
DC.W    $100,$5000
```

Niestety, nie możemy wpisać $8000, bo to oznaczało włączenie wyświetlania w trybie hires!!! Bit odpowiedzialny za 8 bitplanów ma numer 4. Nasza copperlista wygląda więc tak:

```
DC.W    $100,$10
DC.W    $100,$8010 ;
    dla trybu hires
DC.W    $100,$250 ;
    dla trybu super-hires
```

Pozostał jednak jeszcze jeden mały problem. Pamiętamy jak wolno wykonywały się programy umieszczone w pamięci CHIP, gdy wyświetlaliśmy obrazek w hiresie i 16 kolorach? No właśnie. To był kres możliwości kanałów DMA, a komputer pracował dwukrotnie wolniej. W AA-Chipset pojawiła się więc kolejna innowacja. DMA może pobierać teraz długie słowo lub 2 długie słowa za jednym zamachem. Ponieważ stare DMA pobierało tylko jedno słowo, przyspieszenie wyświetlania jest odpowiednio: dwukrotne i czterokrotne. Tylko dlatego możemy cieszyć nasze oczy obrazkami w 256 kolorach w rozdzielczości 1280x512. No cóż, przy tych rozmiarach ekranu, programy z pamięci CHIP zwalniają znów dwukrotnie, ale nie zapominajmy o 14 MHz zegarze.

Pojawia się więc kolejny, nowy rejestr F_MODE ($DFF1FC) do którego wpisujemy, ile kanały DMA mają pobierać danych.

```
DC.W    $1FC,$1 ;
pobierają długie słowo

DC.W    $1FC,$3 ;
pobierają dwa długie słowa
```

Zmieniła się niestety jeszcze jedna rzecz. Załóżmy, że mamy 8 bitplanów ułożonych liniowo (tzn. pierwsza linia pierwszego bitplanu, pierwsza linia drugiego bitplanu, pierwsza linia trzeciego bitplanu, ..., druga linia pierwszego bitplanu, ... W wielu iff konwerterach nazywa się to RAW BLIT, bo jest to wygodne przedstawienie ekranu przy blitowaniu ruchomych objektów. (Wykonujemy wtedy tylko jeden, długi blit, zamiast 8 krótkich). Przy pobieraniu przez kanały DMA tylko jednego słowa ([$DFF1FC] = 0) podalibyśmy modulację równą 7*40. Gdy jednak F_MODE zawiera 1 lub 3 zamiast naszego obrazka ujrzymy efektowną kaszę. Ponieważ mamy jeszcze sprawny umysł z szybką pamięcią, conajmniej 20ns, pamiętamy, że chcieliśmy obejrzeć jakiś obrazek, nie sieczkę. Okazuje się, że modulo musi być pomniejszone o ilość danych, którą pobiera DMA.

```
DC.W    $108,7*40
DC.W    $10A,7*40  ;
gdy (F_MODE) = 0

DC.W    $108,7*40-4
DC.W    $10A,7*40-4 ;
gdy (F_MODE) = 1

DC.W    $108,7*40-8
DC.W    $10A,7*40-8 ;
gdy (F_MODE) = 3
```

Ponieważ zawsze jest coś za coś, dane dla bitplanów muszą być umieszczone od właściwych adresów. I tak:

- adresy bitplanów muszą być podzielne przez 2, gdy (F_MODE)=0,

- adresy bitplanów muszą być podzielne przez 4, gdy (F_MODE)=1,

- adresy bitplanów muszą być podzielne przez 8, gdy (F_MODE)=3.

Cała procedura, która wyświetla obrazek w formacie IFF w rozdzielczości 320x256 i 256 kolorach znajduje się na końcu numeru.

Na koniec jeszcze mała prośba do czytelników o zamiłowaniach koderskich i posiadających na dodatek dowolną Amigę z AA-Chipset (możliwości są dokładnie dwie : 4000 i 1200). Jeśli ktoś z Was odkryje znaczenie jakiegoś nowego rejestru, bądź nowych bitów w starych rejestrach, to prosimy bardzo o kontakt z redakcją. Będziemy wdzięczni i oczywiście opublikujemy Wasze spostrzeżenia za łamach Kebaba.

Robin/WFMH

OGŁOSZENIA DROBNE

Sprzedam programy do C-64. Katalog gratis. (Koperta+znaczek) Jakub Kremiec ul. Powstańców 61. 43-245 Studzionka

Sprzedam podręcznik programowania Simon's Basic. 45 tys. + koszty przesyłki.
Sprzedam książkę "Język maszynowy C64" (2 części) za 75 tys. zł + koszty przesyłki.
Marcin Lis ul. Kochanowskiego 14B/20. 01-864 Warszawa

EXILES (C64) poszukuje grafików, muzyków, koderów! Wymiana programow PUBLIC DOMAIN!
Przemysław Plewa ul. M. Konopnickiej 6/78. 85-124 Bydgoszcz

Opisy programów użytkowych na Amigę. Oferty listowne. Koperta + znaczek. Mieczysław Hałas
ul. Łokietka 12/2 68-100 Żagań

Gry na C64 (kasety) - katalog, znaczek+koperta. Tomasz Samoraj Al. Woj. Polskiego 38/[illegible]
05-800 Pruszków

Sprzedam C64 + data-casette, joy, karta, pokrywa, FINAL II, Black Box. Tanio! Przemysław Bujko, tel 346-74, ul. Wyszyńskiego 6/51. 99-300 Kutno

Sprzedam i zamienię gry i programy na C64. Bogumił Bartyzel
ul. Pułaskiego 95/65
16-400 Suwałki

Grupa "DUET" nawiąże kontakt w celu wymiany oprogramowania i doswiadczeń na C64. Posiadamy dużo nowości. Bartek Uptas, P.O.Box 238
78-100 Kołobrzeg

Sprzedam mapę pamięci C64 na dysku i taśmie. Pełny opis. Cena 40 tys. Andrzej Gruszczyński
ul. Zielonogórska 67/3
66-016 Czerwieńsk

Kupię literaturę i pióro swietlne do C64. Pilnie kupię gry i programy użytkowe na C64 (taśma lub dysk).
Wojtek Zeglin ul. Królowej Jadwigi 20/11, 70-262 Szczecin

Sprzedam gry i użytki na C64 (kasety i dyski). Katalog - koperta + znaczek. Grzegorz Cudny
ul. Powstanców 101/49
05-800 Pruszków

Za stację do C64 (5,25') oferuję ciekawy sprzęt oraz gotowkę.
Mariusz Mincewicz
ul. Przedszkolna 7/24
21-350 Międzyrzec Podlaski

Amiga - sprzedaż oprogramowania, katalogi (koperta + znaczek)
Jacek Ratka ul. Brzozowa 79
13-230 Lidzbark Welski

Sprzedam programy na C64 (dysk,taśma). Posiadam dużo gier i opisów. Samek Bojarski
ul. Mrongowiusza 7
14-100 Ostróda

C64 - gry i użytki (kaseta, dysk). Informacje koperta + znaczek.
Mariusz Listowski ul. Sobieskiego 17/8, 76-200 Słupsk

A2000 karta XT, 2xFDD 1064S, 400 dyskietek, sampler - 18,3 mln.
Marek Dąbrowski ul. Józefowska 114/34, Katowice tel 586-189

Kupię rozszerzenie 1764 do C64 tanio!. Nawiążę kontakt z posiadaczami C64. Roman Iwaniuk
ul. Nowa 8, 11-710 Piecki

Grupa Black Bit (Amiga) poszukuje PILNIE kodera, grafika, muzyka!
Stanisław Zielosko ul. T. Kulika 11
40-424 Katowice 16

Sprzedam C64 (na gwarancji) + magnetofon, gry, pokrywa, joystick, cartridge, literatura. Paweł Paluch tel. [illegible]-14, ul. Wyscigowa 10/[illegible], 26-611 Radom

Rozpocznę korespondencyjną naukę jez. maszynowego (C64).
Dariusz Żukowski, ul. Książąt Opolskich 48/3, 45-005 Opole

Sprzedam nagrane dyski lub skopiuje. Amiga 500/2000. Marek Kosowicz, P.O. Box 1411
Wrocław 16

Sprzedam C64, magnetofon, 33 kasety z grami i użytkami, cartridge: Black Box V.8 i SUPER GAMES. Cena 2.5 mln. Daniel Kukuła
ul. Warszawska 13/46
72-200 Nowogard

Wymienię programy na Amigę 500 oraz doswiadczenia. Krzysztof Leśniewski, ul. [illegible]skiego 52A/[illegible], [illegible] [illegible]ice 3

Mapa pamięci C64. Cena 60 tys. płatne przy odbiorze. (zawsze aktualne), Tomasz Filipowicz
Dąbrowica 26/5
58-500 Jelenia Góra

Automat perkusyjny (MIDI) zamienię na stację dysków do C64.
Dariusz Gawerski, ul. Sportowa 20, 11-200 Bartoszyce

Sprzedam gry, programy i literaturę na C64. Katalog koperta i znaczek. A. Makowski
ul. Będzińska 5, 52-230 Wrocław

Wymienię oprogramowanie (gry i użytki) na C64 (dysk) i Amigę!
Jedrzej Chmielewski, ul. [illegible]wa 7/33, 85-[illegible] Bydgoszcz

Kupię uszkodzoną napęd 3,5' (ew. samą plytkę elektroniki).
Rafał Jaworowski, tel. 867-587
ul. Komandorska 10/15
94-116 Łódź

Sprzedam gry na C64 (kaseta). Wyślę katalog koperta + znaczek.
Piotr Koziel ul. Jaśminowa 7
27-300 Lipsko

Kupię TOP SECRET z lat 90/91 - wszystkie numery. Adam Sawicki
ul. Grochowskiego 23
75-363 Koszalin

Sprzedam C64, stację, 100 dyskietek, monitor, joysticki, cartridge, literaturę. Zbigniew Żulicki
ul. Dworcowa 16
63-325 Kowalew

Sprzedam C64, 1541-II, FINAL III, myszkę, pudełko z dyskami i literaturę (5 mln) lub zamienię na Amigę. (Wymienię programy)
Dariusz Sieradzki, ul. Polna 22/[illegible]
10-[illegible]04 Olsztyn

Szukam programu (nie listingu) DISK SEARCHER (z Bajtka) C64 oraz programów PDP i SHAREWARE. Oferuję wymianę, nawiążę kontakty - C64.
Błażej Strażak, ul. Żarska 20
43-200 Pszczyna

Sprzedam C-128, stacja dysków, joystick, cartridge FINAL II, 100 dysków. Rafał Mierzwa, tel 403-205, ul. Chełmska 24A/11
00-725 Warszawa

Kupię stację 1541 II lub inną do C64. Zamienię C-64II, 1541,turbocorder, 4 joysticki, kasety, literatura na Amigę 500. Przemysław Budzik, ul. K. Makuszynskiego 6
78-100 Kołobrzeg

C-64II + stacja zamienię na Amigę [illegible]0 lub PC AT, Michał Tatarynowicz, ul. 2[illegible] Marca 91D/51
81-82[illegible] Sopot

Kupię FINAL III do 120 tys.
Poszukuję CHAMPION OF THE-RAT (kaseta). Wojciech Szymik
ul.Górnicza 23
44-624 Jankowice

Zamienię C64 z osprzętem, komiksy (150) na Amigę z osprzętem (stan dobry), Tomek Chylicki tel 341-777, al. Piastów 1/7
70-346 Szczecin

Sprzedam C-64II z magnetofonem, blackbox, ponad 100 gier i programów, literatura za [illegible] mln.
Piotr Lewandowski, Os. Zal[illegible] B/1/H/12, 73-110 Stargard

Tanio sprzedam Amiga 500 + stacja + monitor + drukarka + 1MB + modulator. Norbert Papaj
ul. Wiejska 102
58-502 Jelenia Gora

Aparat fotogr. YASHICA G35, nasadki, telephoto + videangle, adapter 35 mm zamienię na stację 1541 II lub inną. Aleksander Szubert, ul. 1 Maja 69/37
95-100 Zgierz

Prosimy kolegę, który napisał do nas w celu nabycia gier do C64 na taśmie, aby przesłał jeszcze raz swój adres. Jacek Krysiak
ul. 1 Maja 4, 88-230 Piotrków Kuj.

TECHNO HOUSE STUDIO AMIGA

Jest to najnowszy zestaw programowy umożliwiający tworzenie prawdziwych utworów techno na poziomie komercyjnym.

Zestaw dysków składa się z:
- **gotowych rytmów**
- **najdzikszych instrumentów techno**
- **specjalnych edytorów do modulowania głosu i dźwięku**
- **specjalizowanego edytora muzycznego.**

Zamówienie proszę składać na adres:
Krzysztof Płonka
ul. Krowoderska 60/5
31-141 Kraków

Sprzedam C-64II, stacja 1541 II, Final III, joystick, dyski, literatura. Cena 3.8 mln. Bartosz Gołębiowski, tel 382-596, ul. Puszkina 2/2, Gliwice

Coder (C64) wstąpi do demo-grupy. (Możliwość kontaktu listownego). Tomasz [illegible]ski
ul. War[illegible]skiego [illegible]
47-223 Kędzierzyn-Koźle

Sprzedam NOWE, także oryginalne programy na C64. Elysium P. O. Box 44
81-106 Gdynia 6

Amiga - programy, niskie ceny, katalog gratis.
Tomasz Chojnacki, tel 431-571, ul. Powstańców Śląskich 57, 32-300 Olkusz

Sprzedam C64, stację dysków, magnetofon, joystick, Final II, oprogramowanie. Dariusz Nowicki tel 534-322, ul. Bandurskiego 59/21, Szczecin

Sprzedam monitor Neptun 156 (zielony, do Amigi lub C64), sampler stereo do Amigi (40 kHz).
Daniel Rastuszko, tel 426-261
ul. Iberyjska 4/2
02-764 Warszawa

LUDZIE! POMOCY!

MOI ZNAJOMI TO SYNONIM KOMPUTEROWEJ PUSTYNI. POSZUKUJĘ KODERA (NAWET POCZĄTKUJĄCEGO), KTÓRY POMÓGŁBY MI W ZGŁĘBIENIU TAJNIKÓW KODOWANIA (A500). PODZIEL SIĘ UMIEJĘTNOŚCIAMI. CEL: ZAŁÓŻMY GRUPĘ I BĄDŹMY NAJLEPSI NA ŚWIECIE!

ARTUR WERNER
ul. Kostrzewy 51,
75-362 Koszalin

Spis artykułów zamieszczonych na łamach Kebaba w 1992 roku.

C64 - Różne

Różne po raz pierwszy

Różne po raz drugi

Gry

Listingi dla C64

Listingi dla Amigi

ZE SCENY

Nie minął miesiąc, a my dostaliśmy od Was kolejną porcję programów. Mamy nadzieję, że dzięki temu rubryka ta zostanie stałą pozycją Kebaba.

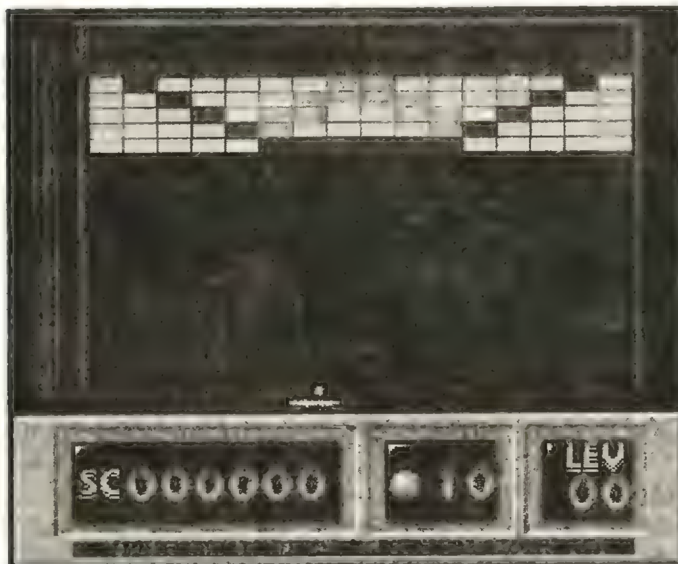
Pierwszą pozycją, jaką chciałbym przedstawić, jest gra napisana przez **Scoby'ego**, o wiele mówiącej nazwie *THE WORMS*. Czy już wiecie o co chodzi? Jeśli tak to posłuchajcie. Gra przeznaczona jest dla dwóch osób, w szczególności dla jednej osoby i komputera (A co! Amidze też należy się czasem chwila rozrywki).



Na ekranie poruszają sie dwa węże (robaki?), z każdą chwilą zwiększające swoją długość. Zadanie polega na takim zręcznym kierowaniu swoim wężem, aby nie wpaść na barierkę, porozrzucane bloki, ani na przeciwnika. Przy powyższych ograniczeniach warto jeszcze tak zajechać drogę partnerowi, aby maksymalnie utrudnić mu życie.

W momencie nieszczęścia, czyli zderzenia się z którymś ze wspomnianych obiektów (inaczej: straty życia), przeciwnik otrzymuje punkty, których ilość jest wprost proporcjonalna do długości węża. Wygrywa ten kto pierwszy uzyska 1000 punktów.

Przed rozpoczęciem gry użytkownik może dość swobodnie "ustawić jej konfigurację": włączyć/wyłączyć dźwięk, ustalić prędkość poruszania się węży, wybrać urządzenie sterownicze (joystick, myszka, klawiatura), lub grę z komputerem. Ten ostatni wariant jest jeszcze niedopracowny - algorytm sterowania wężem daleki jest od doskonałości - proponuję jeszcze nad nim popracować.



Jeżeli zaś zrezygnujemy z gry z naszą Amigą i zagramy z bratem lub babcią, to możemy spodziewać się wielu chwil dobrej zabawy. Gra rozpowszechniana jest na zasadach "freeware", w związku z czym każdy może ją posiadać legalnie, praktycznie za darmo. A oto lista "co kto zrobił" i kontakt z autorami:

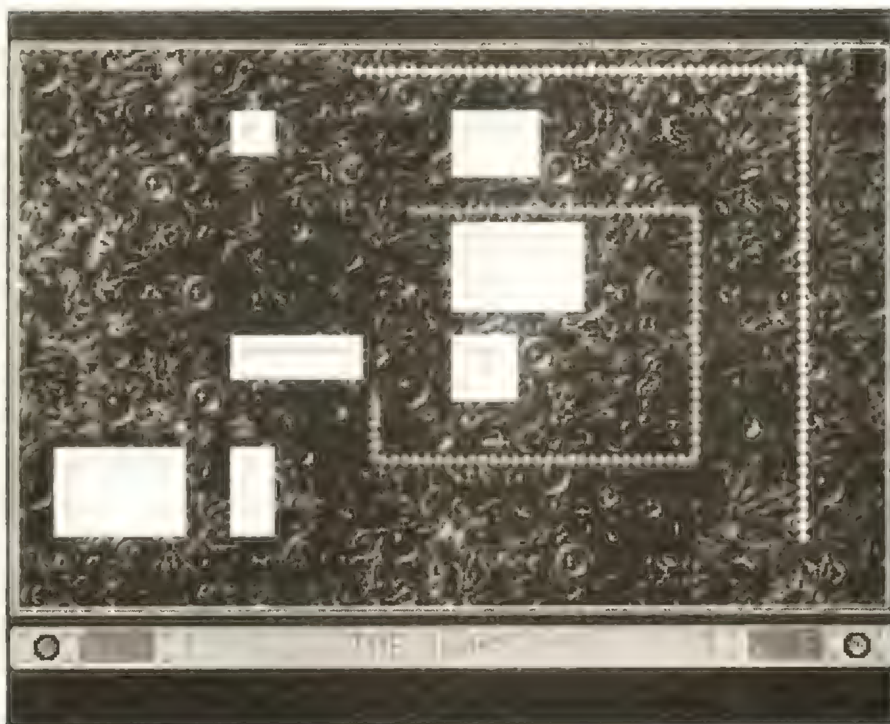
Programowanie: Scoby
Muzyka: J.M.T
Grafika: Scoby

Scoby
ul. Różowa 4/9
70-781 Szczecin

Kolejnym produktem którym mamy okazję się dziś zająć jest music-disk grupy **FUNZINE** o dźwięcznej nazwie *SOUND GARDEN*. Na wstępie muszę przyznać, że miałem pewien kłopot z opisem tego programu albowiem gusta muzyczne są tak zróżnicowane, że to co dla jednego będzie wspaniałym utworem innemu kojarzyć się będzie z przysłowiową "kocią muzyką".

Zatem jako kryterium oceny, z góry zakładając, że nie będzie to w pełni obiektywne, postanowiłem przyjąć własne gusta muzyczne a przy tym starałem się wziąć także pod uwagę aktualne trendy panujące na amigowskiej scenie muzycznej i w rezultacie muszę przyznać, że zawarte w SOUND GARDEN kompozycje, autorstwa **PASSATA**, wypadły bardzo interesująco (przykładowo utwór WINTER 2013,o ile mnie pamięć nie myli wystawiany na Copy-Party w Warszawie, czy też wersja znanego utworu Adamskiego pod tytułem KILLER). Szkoda tylko, że do wysłuchania mamy jedynie sześć utworów.

Niestety nie może się obejść i bez dozy krytyki. Jak na mój gust music-dysk, gdzie jedyną oprawą plastyczną do utworów jest statyczna plansza, to już w pewnym sensie przeżytek. Brakowało mi przykładowo małego equalizera lub choćby nawet efektownej wybieraczki (wybieranie utworu poprzez naciśnięcie odpowiedniego klawisza nie zrobiło na mnie dużego wrażenia).



Ciekawostką są za to wymagania sprzętowe owego music-dysku, który bez rozszerzenia typu SLOW RAM nie bardzo chce działać (Panowie, więcej tolerancji dla biednych użytkowników A500+, A1200 czy miłośników SUPRA RAM!).

Podsumowując, uważam, że SOUND GARDEN, jako ich pierwszy "duży" program stanowi dobrą wizytówkę młodej stażem grupy FUNZINE, zaś usprawiedliwieniem powyższych niedociągnięć niech będzie fakt (co podkreślają sami członkowie grupy), że był on składany w pośpiechu i na monochromatycznym monitorze. Oto autorzy:

Kodowanie: Mac. Ralph
Muzyka: Passat

oraz ich adres kontaktowy:

Mac. Ralph/Funzine
ul. Komandorska 10/15
94-116 Łódź

Na zakończenie prawdziwa rewelacja! Gra nosi nazwę "Double Ball" i została napisana przez przez **MR.GARY'ego** z grupy **Blue Data**. Doskonałe wykonanie techniczne, stawia ten produkt w czołówce gier napisanych przez polskich programistów, a przeznaczonych dla komputera Amiga.

Choć pomysł nie jest niczym nowym, gdyż gra jest adaptacją popularnego Arkanoid'a, "gralność" jest bardzo wysoka. Dla tych, którzy jeszcze nie wiedzą co Arkanoid oznacza, krótki opis.

Całość składa się z 42 plansz, i na każdej z nich musimy zbić przy pomocy piłki i paletki wszystkie klocki. Gdy tego dokonamy przechodzimy do następnego etapu. Podczas gry, po zbiciu specjalnych klocków czekają nas różne niespodzianki: dodatkowe życie, extra punkty, poszerzenie paletki, dodatkowa piłka, czy przejście do następnego etapu.

Gra nie jest zbyt łatwa (szczególnie dla takiego niedzielnego gracza jak ja) i choć 10 "żyć", które mamy na początku do dyspozycji wróży szybkie jej ukończenie, to po pierwszym aktywnym kontakcie z nią okazuje się, że będziemy musieli spędzić przed komputerem kilka wieczorów, aby ujrzeć to co znajduje się po ostatnim etapie.

64

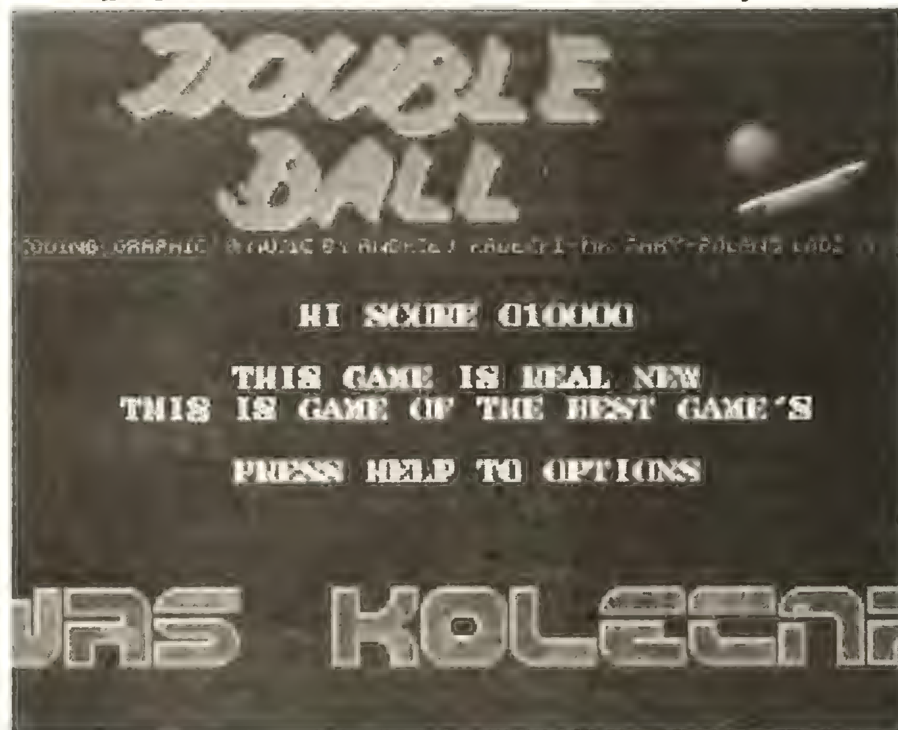
Oprócz gry MR. GARY przysłał również do redakcji program muzyczny, oraz konwerter do grafiki własnego autorstwa. Niestety programy te miały kłopoty z systemem 2.0, co w dzisiejszych czasach już nie powinno mieć miejsca.

Oto adres autora:

MR. GARY
Andrzej Radecki
ul. Gorkiego 17m193
92-525 Łódź

Niezmiernie nam miło, że w naszym kraju coraz więcej osób próbuje swoich sił w programowaniu i to z takimi skutkami. Miło nam tym bardziej, że to właśnie na łamach Kebaba możemy zaprezentować ich pierwsze prace. Mamy nadzieję, że na tym nie skończy się Wasza współpraca z nami. Czekamy na dalsze programy zarówno na C64 jak i Amigę.

Krzysztof Kobus



Zapomniane urządzenia cz. 1.

Dyskietka zatytułowana "Workbenchx.x" jest zazwyczaj pierwszą, którą wkładamy do stacji dysków naszej nowo zakupionej Amigi. W większości przypadków, dyskietka ta ląduje następnie w najdalszym zakątku naszej szuflady, lub wszystkie dane na niej się znajdujące ustępują miejsca nowo zakupionej (ukradzionej?) grze.

Ten artykuł jest dla tych wszystkich, którzy pomimo wszystko posiadają jeszcze oryginalną zawartość owej dyskietki, i chcieliby poszerzyć swoją wiedzę na temat funkcjonowania pewnych mechanizmów dyskowego systemu operacyjnego Amigi, zwanych popularnie urządzeniami.

Włóżcie zatem "Workbenchx.x" do stacji dysków i jednocześnie wciśnijcie klawisze "CTRL" oraz "D", trzymając je tak długo, aż pojawi się napis

**BREAK - CLI

lub

**BREAK - SHELL

W tym momencie przerwane zostało wykonywanie "startup-sequence". Użytkownicy systemu 2.0, lub wyższych mogą w tym samym celu posłużyć się znacznie bardziej elegancką metodą, mianowicie w momencie resetowania, bądź włączania komputera proszę przytrzymać wciśnięte oba przyciski myszy.

Powinien pojawić się ekran tzw Boot-Menu, a na jego dole przycisk "Advanced Options", który to przyciskamy, następnie przełącznik przy napisie "Startup-sequence" ustawiamy w położenie "DISABLED", co uniemożliwi systemowi wykonanie sekwencji startowej.

Wychodzimy z ekranu opcji zaawansowanych (skoro już z nich skorzystaliśmy i umiemy wyłączyć startup-sequence to bez obaw możemy ogłosić wszem i wobec, że jesteśmy profesjonalistami), wskazujemy przycisk "DF0:", po czym komputer zakończy proces "bootowania" i... ignorując startup-sequence z katalogu "S" będzie czekał na nasze polecenia.

Właściwie znaleźliśmy się znów w punkcie wyjścia, i zapewne (nota bene całkiem słusznie) zauważacie, że nadszedł czas na poznanie pierwszego urządzenia. Proszę jednak jeszcze o chwilę cierpliwości i wyrozumiałości, gdyż chciałbym wyjaśnić jeszcze pojęcie **standardowego wejścia** (stdin) i **standardowego wyjścia** (stdout).

Zacznę może od tego drugiego. Otóż standardowe wyjście jest to urządzenie do którego programy wysyłają komunikaty, lub wyniki określonych operacji. Weźmy na przykład komendę "dir". Wynikiem jej wykonania jest lista plików i podkatalogów katalogu aktualnego (o ile po "dir" nie podaliśmy żadnego parametru).

Lista ta zostanie wypisana, jak wynika z przytoczonej wyżej definicji, do standardowego wyjścia, a że to po zresetowaniu komputera jest automatycznie ustawiane na okienko CLI, to właśnie w nim będziemy mogli ją odczytać. I nie byłoby w tym nic godnego uwagi, gdyby nie jeden drobny fakt, że standardowe wyjście możemy dowolnie przełączyć. Jak? Używając jednego tylko znaku większości: ">" i nazwy nowego "stdout". Napiszmy zatem:

```
dir >ram:spis
```

Stacja dysków zakręci się parę razy i... żadnego efektu. Czyżby? Proszę napisać:

```
dir ram:
```

Na ramdysku pojawił się tajemniczy plik o nazwie "spis". Sprawdźmy jego zawartość. Tutaj możemy posłużyć się dowolnym edytorem tekstu, lub komendą `type`. Skorzystajmy z tego drugiego sposobu.

```
type ram:spis
```

Na ekranie pojawiła się zawartość pliku tekstowego "ram:spis", utworzonego po wydaniu komendy `dir >ram:spis`.

Jak widzicie plik ten zawiera te same informacje, które uzyskaliśmy przy pomocy samej komendy `dir`. Zatem różnica między tymi polega jedynie na sposobie wyprowadzenia wyników działania komendy, w tym wypadku listy zawartości aktualnego katalogu. Sekwencja ">ram:spis" spowodowała czasowe (na czas wykonania komendy "dir") przełączenie standardowego wyjścia będącego okienkiem CLI, na plik "ram:spis".

W związku z tym wszystkie komunikaty pochodzące od komendy, nie zostały wyprowadzane do CLI, tylko do wspomnianego pliku "spis". Ogólnie mówiąc przełączenie standardowego wyjścia realizujemy wpisując nazwę komendy, zaraz za nią po spacji znak ">", nazwę nowego "stdout", a dopiero następnie ewentualne inne parametry dla komendy. Jeśli zatem zawartość katalogu "df0:Prefs" będziemy chcieli umieścić w pliku "ram:temp", to uzyskamy to wydając komendę:

```
dir >ram:temp df0:prefs
```

Niepoprawnym natomiast jest zapis:

```
dir df0:prefs >ram:temp
```

chyba, że posiadamy system operacyjny OS 2.0 lub wyżej, który dopuszcza również taką składnię.

Do zilustrowania powyższych przykładów użyłem komendy "`dir`". Oczywiście analogicznie zachowują się wszystki inne komendy z katalogu "C", oraz wszystkie inne progra-

my wypisujące dowolne komunikaty w okienku CLI.

O standardowym wyjściu można napisać jeszcze wiele. Nie to jest jednak tematem naszego artykułu, przejdę zatem do omówienia standardowego wejścia. Pojęcie to nie wydaje się być tak oczywiste, jak to omawiane porzednio. Niewiele bowiem komend w sposób jawny prosi nas o podanie danych wejściowych, gdyż zazwyczaj parametry dla komend podajemy w momencie ich wywoływania.

Najprościej rzecz biorąc można powiedzieć, iż "stdin" jest miejscem z którego odczytywane są parametry dla programu, po jego uruchomieniu. Z powodu braku komendy, która pozwoliłaby najprzejrzyściej zilustrować tę definicję przykładem, wymyślmy sobie program o nazwie "Dodaj" proszący nas o podanie jednej liczby, potem drugiej, a następnie wyświetlający ich sumę. Jeżeli teraz uruchomimy go pisząc:

```
Dodaj
```

Pojawi się komunikat proszący nas o podanie pierwszej liczby - wpisujemy na przykład 2, następnie w odpowiedzi na prośbę o podanie drugiej liczby wpisujemy 3. W wyniku naszych działań w standardowym wyjściu, czyli okienku CLI powinna ukazać się liczba 5 będąca wynikiem działania programu "Dodaj". W tym wypadku możemy powiedzieć, że liczby 2 oraz 3 zostały wprowadzone do programu ze standardowego wejścia, a tym aktualnie było okienko CLI. Gdyby zaś specyfika programu zmuszała nas do podania dwóch liczb (jako jego parametrów) w momencie uruchamiania go, na przykład w następujący sposób:

```
Dodaj 2 3
```

to nie moglibyśmy powiedzieć, że owe liczby zostały wprowadzone ze standardowego wejścia. Stąd właśnie konieczność odwołania się do tego wyimaginowanego przykładu (komendy napisane są w ten sposób, że parametry musimy podawać w momencie ich uruchamiania, jeżeli tego nie uczynimy to ujrzymy komunikat "Bad arguments").

Stdin ma to do siebie, że podobnie jak "stdout" może zostać swobodnie przełączane.

Dokonujemy tego przy pomocy znaku mniejszości "<". Sporządźmy zatem przy pomocy edytora tekstów plik o nazwie "ram:Dane", którego zawartość jest następująca:

```
2
3
```

czyli w pierwszej lini liczba 2 (RETURN), w drugiej liczba 3 (RETURN).

Zatem plik powinien mieć 4 bajty długości. Teraz proszę wykonać:

```
Dodaj <ram:Dane
```

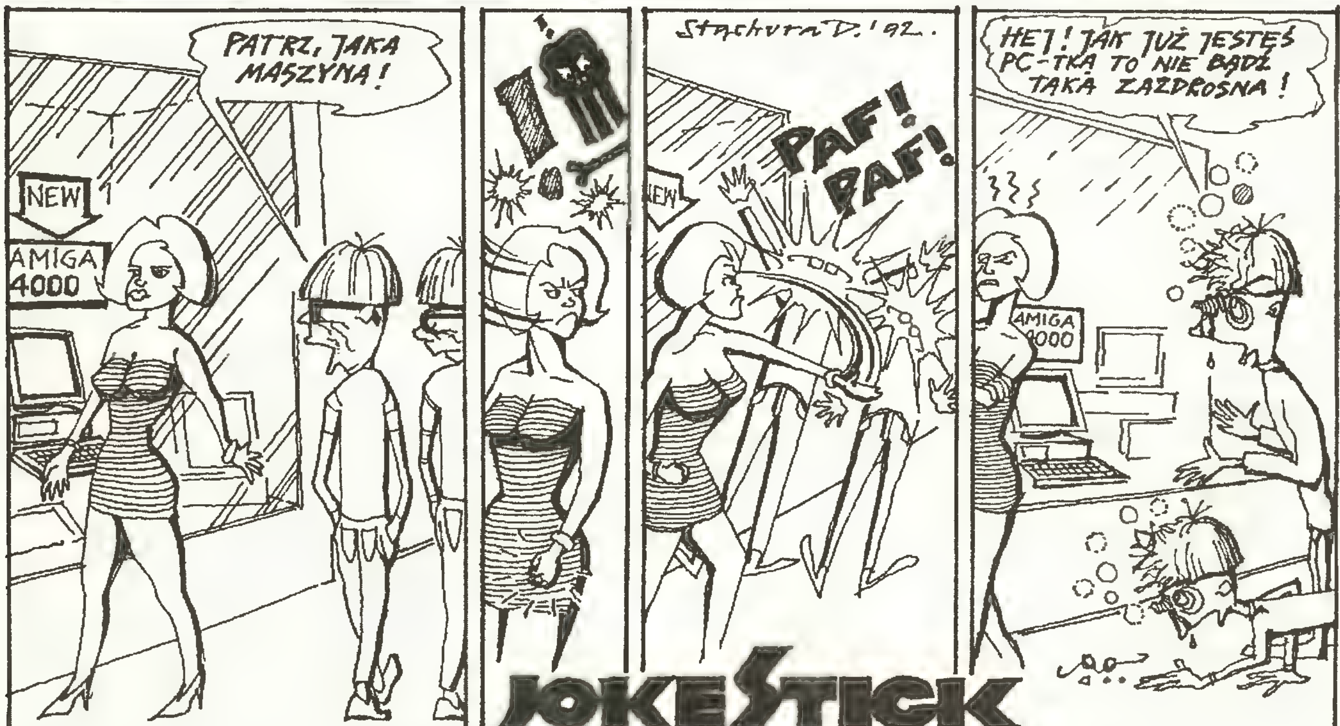
Tym razem program również wypisze komunikaty proszące nas o podanie liczb, jednak nie będzie czekał do momentu aż wprowadzimy je z klawiatury, tylko pobierze je z plku "ram:Dane", gdyż na niego właśnie zostało przełączone standardowe wejście. Oczywiście również poprawnym jest następujące wywołanie programu:

```
Dodaj >ram:Wynik <ram:Dane
```

Jego wynikiem będzie utworzenie pliku "ram:Wynik", zawierającego komunikaty proszące o podanie liczb wejściowych, a także wynik dodawania, czyli liczbę 5. Dane wejściowe zostaną oczywiście pobrane z pliku "ram:Dane".

No cóż, miało być o urządzeniach a było o stdin i stdout. Zapraszam zatem do lektury za miesiąc, gdzie na pewno znajdziecie drugą część tego artykułu - zgodną merytorycznie z jego tytułem.

Krzysztof Kobus



Tutaj chciałbym jeszcze skierować parę słów do osób zajmujących się programowaniem. Mogą bowiem pojawić się wątpliwości jak należy pisać programy, aby mogły one wykorzystywać opisane tu mechanizmy. W przypadku zwolenników języka C, sprawa jest całkiem prosta, gdyż wszystkie funkcje służące do komunikacji komputera ze środowiskiem zewnętrznym, a mające swoje prototypy w nagłówku <stdio.h> (np. gets(), printf(), scanf()), tak na prawdę służą do komunikacji nie z ekranem, bądź klawiaturą, ale właśnie z stdin/stdout. W przypadku assemblera, sprawa wygląda podobnie.

Aby coś wydrukować w okienku CLI, należy przy pomocy funkcji Output() znaleźć handler, który potem należy podać właściwej funkcji w tym wypadku drukującej Write(). Jak można się spodziewać wspomniany handler nie dotyczy okienka CLI, ale właśnie standardowego wyjścia. Analogicznie sprawa wygląda w przypadku standardowego wejścia i funkcji Input(). Wynika z tego, że każdy program korzystający z Input(), bądź Output() będzie współpracował z opisywanymi tu mechanizmami. Na zakończenie dodam jeszcze, że wzmiankowane tu funkcje języka C, również korzystają z Output(), i Input().

Packet Radio

- odc. 1

Co to jest Packet Radio... nie każdy wie! Musimy zatem powiedzieć coś więcej. Packet Radio (dalej nazywane w skrócie: PR) jest to jedna z cyfrowych metod przekazywania informacji na odległość. Metoda ta jest z powodzeniem stosowana w łącznościach radioamatorskich (tych prawdziwych tj. licencjonowanych) oraz w tzw. paśmie 11m ogólnie znanym jako CB.

O korzeniach PR mieliśmy już okazję pisać w KEBAB'ie. Dzisiaj zajmiemy się stroną bardziej praktyczną całego zagadnienia. Wiemy już (z wcześniejszego KEBAB'a), że na potrzeby PR został zaadaptowany komercyjny protokół przesyłu informacji o nazwie X.25.

Po pewnych modyfikacjach uzyskał on nazwę "AX.25", gdzie "A" oznacza (A)matorski. Całość opiera się na kodowaniu informacji w postaci "Packet'ów". Słowo "Packet" można przetłumaczyć jako "pakiet". Oprócz tego spotyka się również często w literaturze angielskie słowo "frames" (po polsku: ramki). Właściwie chodzi o jedno i to samo. Informacje trzeba "upchnąć" w odpowiednie pakiety. Jeżeli stacja A ma zamiar powiedzieć do stacji B np. "Sie ma" to wysyła odpowiedni pakiet, w którym będzie zakodowane (zgodnie z AX.25) odpowiednio kto jest nadawcą danego pakietu, kto ma być jego odbiorcą, sama właściwa informacja (w naszym przypadku: "Sie ma") a także odpowienie sumy kontrolne i inne informacje służące prawidłowemu przekazowi naszego krótkiego pozdrowienia.

Zaraz! Stacja wysyła? Ale jak? No właśnie! Zapomnieliśmy powiedzieć co nam jest potrzebne. Potrzebne nam jest urządzenie nadawczo-odbiorcze... Mocno powiedziane! Ale nie przerażajmy się! Aby sobie "popaketować" wcale nie będziemy musieli włamywać się np. do lokalnej stacji telewizyjnej.

Wystarczy nam jakiekolwiek proste urządzenie radioamatorskie np. takie jak na zdjęciu. Oczywiście warunkiem korzystania z takiego urządzenia jest posiadanie licenji radioamatorskiej. Jeżeli natomiast takowej nie posiadamy to musimy zaopatrzyć się w radyjko "sibi" czyli "urządzenie nadawczo-odbiorcze przeznaczone do pracy w paśmie 11m".

Co prawda prowadzenie łączności (a więc również PR) jest w tym paśmie zdecydowanie trudniejsze ze względu na wysoki poziom zakłóceń oraz brak konkretnych (pisanych) reguł odnośnie korzystania z poszczególnych kanałów, to niemniej jednak efekty mogą być w zupełności zadowalające.

Jeżeli mamy już sprawy radiowe załatwione, to możemy zastanawiać się co dalej będzie nam potrzebne. Prawda! Komputer! Komputer jest do tej całej zabawy absolutnie niezbędny. Zakładam, że komputer każdy z zainteresowanych już posiada. Teraz pojawia się największy kłopot.

Co by tu zrobić, żeby to jedno (radio) połączyć z tym drugim (komputerem) w jakąś sensownie działającą całość? Przykłady na to mieliśmy już w KEBAB'ie. Wtedy zajmowaliśmy się alfabetem Morse'a. Tym

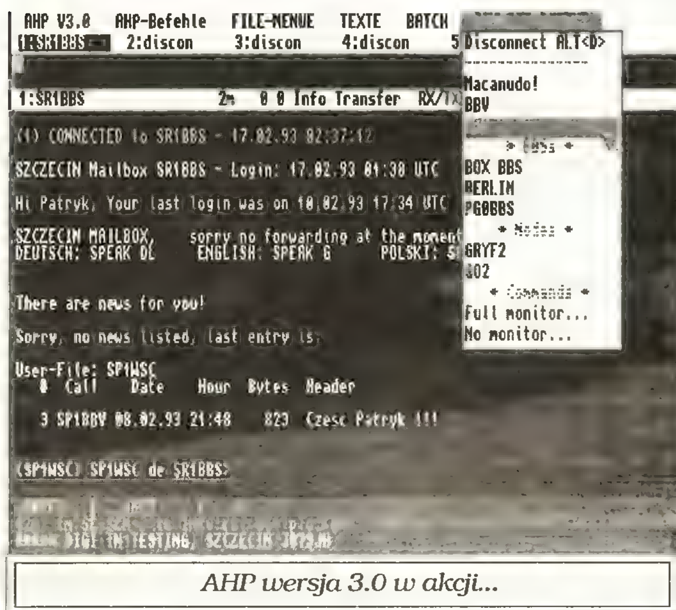


razem jednak chcemy zabrać się za PR. Musimy zatem wyjaśnić sobie pewne zasady.

Jeżeli my, jako w/w stacja A chcemy wysłać w/w stacji B naszą w/w wiadomość, to chcielibyśmy zrobić to jak najprościej. Najlepiej gdybyśmy mogli napisać to na swojej klawiaturze, a na ekranie podłączonym do komputera podłączonego do odbiornika stacji B, pojawił się właściwy napis. Żeby tak mogło się stać potrzebny nam jest odpowiedni program do naszego komputera.

Program ten odczyta to, co napiszemy z klawiatury. Następnie informacja ta musi zostać gdzieś zakodowana w postaci odpowiedniego pakietu zgodnie z protokołem AX.25. Kolejną czynnością będzie włączenie nadajnika i na końcu, dołączony do komputera (i radia), modem wyśle wspomniany pakiet w eter jako charakterystyczne "brzzzzyrzzyt". Na tym przykładzie widać ile mamy etapów zanim nasze "sie ma" zostanie tylko wyemitowane z naszej anteny.

Po drugiej stronie, tj. w stacji B, musi nastąpić proces odwrotny. Odebrane przez odbiornik "bzz...", musi zostać przez modem zdekodowane, następnie otrzymany pakiet musi zostać rozkodowany (i sprawdzony) zgodnie z zasadami AX.25. Jeżeli po rozkodowaniu okaże się, że pakiet dotarł bezbłędnie, tzn. zgadzają się sumy kontrolne i jest adresowany do tej akurat stacji, to wydobyta z niego informacja jest przekazywana dalej do programu, który wyświetli ją na ekranie komputera stacji B.



AHP wersja 3.0 w akcji...

Najtrudniejszy w tym całym łańcuszku, jest etap kodowania i dekodowania pakietów w/g protokołu AX.25. Zawsze pomiędzy naszym komputerem a modemem (i dalej radiem) musi występować coś, co przełoży nam "z komputerowego na pakietowe" i odwrotnie. W zasadzie istnieją dwie metody zainstalowania takiego "tłumacza".

Jedną z nich jest zakupienie (bądź zrobienie) specjalnego urządzenia zwanego TNC (patrz zdjęcie). Jest to skrót od angielskiego: Terminal Node Controler. Urządzenie takie zawiera w sobie zarówno modem, jak i niewielki komputer z własną pamięcią ROM i RAM. Dzięki takiemu rozwiązaniu, pakiety docierające do naszego odbiornika są początkowo (po zdekodowaniu przez modem) przechowywane w RAM'ie TNC.

Następnie procesor TNC, korzystając z własnego, zwartego w ROM'ie programu, dokonuje w/w tłumaczeń w obie strony. Komunikacja na linii komputer TNC, odbywa się poprzez interface RS232. Dzięki zastosowaniu TNC, program pracujący w naszym komputerze musi jedynie przesyłać i odbierać dane z TNC nie martwiąc się o cały, zawiły protokół AX.25. Programy współpracujące z TNC mogą być, w najprostszym przypadku, zupełnie prościutkimi programikami typu "terminal". Na C-64 załatwia nam sprawę kilka linijek BASIC'a.

Na Amigę jakikolwiek program komunikacyjny. Oczywiście istnieją programy bardziej zaawansowane, które współpracują z TNC w tzw. trybie Host-Mode. Przykładowo AHP, SDP na Amigę lub SP, THP na smutnych (iBM'ach). Pamiętać natomiast trzeba, że C-64 jest wyposażony w złącze RS232C o niestandardowych poziomach sygnałów.

W związku z tym należy jeszcze wykonać prosty interface w oparciu np. o układ MAX232 (w przypadku gdy kupiliśmy TNC) lub tak zaprojektować własny TNC, aby poziomy logiczne odpowiadały standardowi TTL. Wadą takiego rozwiązania jest dosyć wysoki koszt samego TNC. Zalety? Wszystkie pozostałe. Innym rozwiązaniem "tłumacza" protokołu AX.25 jest zastosowanie takiego programu, który oprócz samych funkcji terminalowych zajmie się także stroną samego protokołu.

Najlepszym przykładem takiego programu jest sławny Digi-Comm na C-64. Program ten ulepszany kolejno przez różnych radioamatorów osiągnął aktualnie bardzo wysoki stopień zaawansowania. Krążą nawet pogłoski o dokonaniu konwersji tego programu na potrzeby Atari 800 i jemu podobnych.

Dla Amigi i iBM'ów istnieją specjalne "nakładki" przechwytujące kontrolę nad portami komunikacji serial i spełniające funkcje TNC za wyjątkiem modemu. Dzięki tym nakładkom, możliwa jest praca programów przeznaczonych do współpracy z TNC tak, jakby faktycznie był on do komputera podłączony. Zastosowanie takiego "software'owego" rozwiązania pozwala na znaczne obniżenie kosztów.

Programy do PR, są zazwyczaj tzw. HAMware, czyli nie trzeba za nie płacić a wystarczy jedynie przekazywać innym użytkownikom (radioamatorom) w niezmienionej postaci. Ewentualna opłata dla autora wiąże się najczęściej z możliwością otrzymania drukowanej instrukcji i następnej, ulepszonej wersji. Od strony sprzętowej pozostaje nam do wykonania stosunkowo prosty modem. Schemat takiego urządzenia zostanie wkrótce opublikowany w KEBAB'ie.

Jeżeli uda nam się wszystko skompletować, to pozostaje nam już tylko znaleźć jakiegoś podobnego maniaka, z którym będziemy mogli się komunikować. O ile w amatorskich pasmach UKF (np. 2m) nie powinno to stanowić specjalnego prob-

lemu, o tyle w 40-tu kanałach przeznaczonych do łączności CB, mogą pojawić się pewne kłopoty. Znane, światowe grupy Packet Radio działające w tym paśmie, mają swoje stacje ustawione na nieco innych (w Polsce niedopuszczonych do użytkowania) częstotliwościach.

Oprócz tego do transmisji w tym paśmie używa się najczęściej tzw. modulacji LSB. W najprostszych "radyjkach" niestety nie ma możliwości korzystania ani z innych modulacji niż AM/FM, ani z innych niż podstawowe 40 kanałów, częstotliwości. Oczywiście, jeżeli posiadamy sprzęt radiowy nieco wyższej klasy to możemy połączyć się np. ze Szczecina do Sztokholmu poprzez... Neapol we Włoszech. Tak! Packet Radio umożliwia nie tylko łączności pomiędzy dwoma stacjami jak w przypadku klasycznej komunikacji modemem telefonicznym.

Każda napotkana stacja może być nie tylko adresatem przesyłanych do niej informacji, ale również przekaźnikiem informacji adresowanych do innej stacji. Protokół AX.25 dopuszcza maksymalnie osiem stacji pośrednich. Oprócz tego każda stacja może prowadzić jednocześnie kilka łączności na różnych "kanałach logicznych" mimo nadawania i odbioru w jednym tylko kanale fizycznym tzn. na jednej częstotliwości.

Również inaczej wygląda praca ze stacjami typu BBS. Ponieważ do jednego BBS'a może być jednocześnie podłączonych wielu użytkowników, nie ma takiego problemu jak z niektórymi profesjonalnymi BBS'ami np. w naszej Stolicy, obsługiwanymi jedną tylko linią telefoniczną, gdzie użytkownik jest wyrzucany po upływie... sześciu minut.

Tak więc komunikacja przy wykorzystaniu PR wydaje się być, w wielu przypadkach, bardziej pasjonująca niż standardowa KTK wykorzystująca linie telefoniczne. Nie do pominięcia jest również fakt, że za korzystanie z "łączy eterowych" nie musimy nic płacić ponad standardowe opłaty dla PAR'u. Do dzisiaj krąży wśród "modemowców" anegdota o pewnym fanatyku komputerowej telekomunikacji (KTK), który chciał znaleźć się w księdze rekordów Guinessa, jako człowiek, który dokonał transmisji danych na największą odległość i... znalazł się w tejże księdze jako człowiek o najwyższym na świecie rachunku telefonicznym!!! Zapraszam niniejszym już za miesiąc na drugą, bardziej już techniczną część naszych rozważań na temat Packet Radio.

SD!

P.S. Ewentualne uwagi i sugestie od działających kolegów "Packetowców" można kierować również na adres:

SP1WSC @ SR1BBS (2m, 70cm)



Gra wykonana jest przez stosunkowo młodą grupą "TEAM-17", która mimo niewielkiego stażu zdobyła już sobie wielu sympatyków. Czy pamiętacie ich pierwszy przebój "FULL CONTACT", ze wspaniałym animowanym intrem, swietną grafiką i muzyką ?

Czytelnik, który regularnie czyta "KEBABA ", z pewnością zauważył, że większość gier opisywanych na naszych łamach , wymaga od gracza chociaż trochę wysiłku umysłowego. Tym razem jednak postanowiliśmy pozwolić odpocząć Waszym szarym komórkom i dla odmiany przygotowaliśmy coś innego, co wcale nie znaczy, że gorszego, mianowicie grę zaliczaną do gatunku określanego jako "shoot'em up". Ktoś mógłby w tym momencie powiedzieć: "kolejna strzelanina". Owszem, strzelanina, ale za to w jakim wykonaniu !

Nie należę do ludzi którzy zachwycają się pierwszą lepszą grą, lecz **PROJECT X** zrobił na mnie ogromne wrażenie.

Wprawdzie przyjemność grania (playability) była nieco mniejsza, lecz jak na początek, zrobili kawał dobrej roboty. Następnym prduktem TEAM'u był "ALIEN BRRED", który w zainteresowanych kręgach także nie przeszedł bez echa (kto nie grał niech żałuje).

Powróćmy jednak do "PROJECT X". Darujemy sobie wszelakie historyjki (w stylu: zły władca Imperium porwał ... a Ty w odwecie musisz ...., jednak nie możesz zawieść albowiem ... (wykropkowane miejsca każdy wypełni według uznania)), które zwykle załączane są do gier, albowiem sens ich istnienia jest w tym przypadku raczej znikomy. Atutem tej gry jest bowiem bardzo dobra grafika (m.in świetne screen'y pomiędzy poszczególnymi poziomami) i całoekranowa animacja oraz kilka efektów, niespotykanych w grach tego typu (np. samplowane "komentarze") oraz, a może przede wszystkim, szybka akcja.

Po wczytaniu, naszym oczom ukaże się obszerne, w porównaniu z innymi grami typu shoot'em up, menu:

ONE PLAYER - przyciskiem fire decydujemy o ilości graczy (1 lub 2), szkoda tylko, że nie przewidziano jednoczesniej gry dwóch graczy.

START LEVEL - po stracie wszystkich "żyć", możemy wybrać sobie etap od którego rozpoczynamy kolejną grę. Jednak warunkiem startu np. od etapu drugiego, jest wcześniejsze zakończenie gry właśnie na nim (opcja ta, nie działa przy "rookie mode" - patrz dalej).

OPTIONS:

AUTO-FIRE ON-OFF - (raczej nie wymaga komentarza)

LOAD i SAVE HISCORE - nagrywanie/załadowanie tabeli najlepszych wyników (na którą należy przeznaczyć osobny dysk !)

ARCADE MODE - normalny tryb gry.

ROOKIE MODE - wariant gry z pewnymi ułatwieniami tzn:

* mniejsza odporność atakujących

nas pojazdów na nasze strzały (dlatego nie zdziwcie się, gdy pojazd który przyjdzie wam zniszczyć będzie już trochę pogruchotany).

* mniejsza częstotliwość strzałów wymierzonych w naszym kierunku.

* po każdym zniszczeniu naszego statku, broń jest słabsza tylko o jeden punkt.

(Jeżeli ktoś z Was zdecyduje się na ROOKIE MODE, niech raczej nie liczy na wpis.)

RAVE MUSIC MODE - wybór muzyki. Niestety nie będzie nam towarzyszyć podczas właściwej gry.

SELECT CRAFT - chyba najważniejsza z opcji. Po wciśnięciu fire będziemy musieli się zdecydować na jeden z trzech dostępnych kosmicznych pojazdów (wyboru dokonujemy odpowiednim ruchem joysticka, zaś zatwierdzamy przyciskiem fire). A oto co stoi do naszej dyspozycji:

CRUX II (PANCERNIK)

ZALETY - bardzo dobre uzbrojenie. Praktycznie dostępna jest dla niego każda broń - maksymalna możliwość rozbudowy i wzmocnienia arsenału

WADY - mała szybkość - bardzo duża bezwładność, szczególnie przy mocniejszym uzbrojeniu

OCENA: Jestem pewien, że każdy początkujący gracz wybierze właśnie ten statek, sugerując się dużą siłą ognia. Zapewniam Was, że nie jest to najlepsza decyzja. Jeżeli chodzi o szybkość to można sobie poradzić (przez użycie na początku gry opcji SPEED), gorzej natomiast z bezwładnością. Jest tak duża, że dokładne sterowanie wymaga od grającego bardzo dobrego refleksu i przecyzyjnych ruchów joystickiem.

HYPERION (KRĄŻOWNIK)

ZALETY - całkiem dobry arsenał - dobra szybkość

WADY - bardzo słaba broń na starcie - dosyć duża bezwładność (jednak nie tak wielka jak w przypadku "CRUX`a II")

OCENA: Ogólnie bardzo dobry. Jednak ze względu na słabą broń przy starcie, radzę już na początku użyć opcji GUNS, w innym przypadku ewentualne zetknięcie z przeciwnikami skończy się dla was tragicznie.

HUNTER MK 7 (MYŚLIWIEC)

ZALETY - bardzo duża szybkość - minimalna bezwładność - świetnie reaguje na ruchy joystick'a

WADY - ograniczona możliwość rozbudowy uzbrojenia

OCENA: Moim zdaniem jest on najlepszym ze statków. Ograniczenia dotyczące broni nie będą nas dotyczyć jeśli będziemy rozbudowywali "plazmę", która dla "HUNTER`a" jest wręcz konieczna. Nie przesadzajcie też z szybkością !

Po wyborze statku możemy wreszcie rozpocząć grę. Po dokładnym zilustrowaniu ekranu gry, spostrzeżecie zapewne u dołu 9 opcji, którę umożliwiają rozbudowę uzbrojenia:

SPEED - zwiększa szybkość pojazdu.

GUNS - jej zaletą, pomimo tego iż jest to najsłabsza broń ze wszystkich, jest możliwość rażenia pod kątem i to zarówno w górę jak i w dół SIDE - świetna jako osłona z góry i z dołu (bez tego daleko nie zalecisz)

MISSILE - rakiety samosterujące,. Warto je brać ponieważ (podobnie jak w przypadku SIDE), nie tracimy aktualnie używanej broni. Wadą jest zwiększenie bezwładności naszego pojazdu.

PLASMA - moim zdaniem najlepsza broń z oferowanego uzbrojenia. Jej prawdziwe atuty ujawniają się jednak dopiero po "rozbudowie"

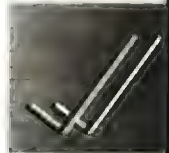
MAGMA - właściwie nadaje się tylko dla "CRUX`a II", ponieważ żaden inny statek nie ma możliwości jej tak wielkiego rozbudowania jak pancernik.

LASER - bardzo silna broń, jednak o zbyt wąskim polu rażenia a zatem niezbyt skutecznie chroniąca przód statku.

STEALTH - jeśli jakiś fragment gry wydaje siś Wam zbyt trudny, warto z korzystaś z tej opcji, albowiem czyni ona Wasz statek na jakiś czas "nieśmiertelnym"

Jak wykorzystać opisane wyżej opcję ? Każdorazowe "zebranie" literki **P** (power), powoduje podświetlenie kolejnej opcji. Jeśeli dana opcja (np. PLASMA) Ci odpowiada to jej wybór potwierdzasz wcisnieciem spacji (jeśli ktoś nie chce odrywać rąk: dwoma szybkimi ruchami joystick`a w lewo) co jest potwierdzone przez komputer głosem. Szary kolor danej opcji informuje nas o wyczerpaniu możliwości jej używania.





Na zakończenie krótka charakterystyka poszczególnych etapów:

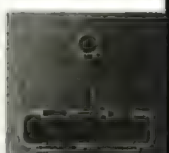
ETAP 1 - jest stosunkowo łatwy. Nawet niedoświadczony gracz po krótkim treningu powinien go ukończyć. Początkowo dużą trudność mogą sprawić rakiety samosterujące, które atakują jednocześnie z góry i z dołu. Radzę wtedy trzymać się przy końcu ekranu, lub skorzystać z opcji STEALTH.

ETAP 2 - Odpowiednio trudniejszy (co chyba oczywiste). Dodatkową przeszkodą są skały i drzewa które raczej należy omijać.

ETAP 3 - Jeśeli ktoś z Was go ukończy (bez pomocy trainer'a), może być z siebie dumny. Etap ten wymaga bowiem od grającego, znakomitego refleksu i dużego opanowania. Z atakujących nas na końcu tego etapu dwóch "wariatów" wystarczy strącić jedenego (będę kłaniał się na ulicy każdemu, kto rozszyfruje schemat ich poruszania się).

ETAP 4 - Częściowo jest on zalany wodą, która czasami znacznie utrudnia pilotowanie gwiezdnego pojazdu. Połowa sukcesu w tym etapie to delikatne ruchy joystick`a.

ETAP 5 - Etap ostatni. Podpowiem ,że do kończącego ten etap pojazdu trzeba się zabrać w nietypowy sposób (ale nie zdradzę jaki). Ponadto obiekty które będą nas atakować są bardzo odporne na nasze strzały, dlatego też, jeśli do tego etapu dolecieliście z 'cieniutkim' uzbrojeniem to radzę raczej wyłączyć komputer i zagrać w bierki. Wysadzenie bazy (celne oko zalecane) wieńczy dzieło, pozwalając spokojnie oczekiwać na "PROJECT Y".



Ponadto, po ukończeniu drugiego etapu, każdy następny poprzedzony jest ciekawie zrobionym etapem 'bonusowym'. Good Luck (i niech Moc będzie z Wami).

Jarosław "OBCY" Gumowski

P.S. Gra zajmuje cztery dyski oraz wymaga minimum 1 MB pamięci RAM.

Champion 2

Ten artykuł chciałbym poświęcić wspaniałej grze fantasy, składającej się moim zdaniem na klasykę gier na Amigę. Do napisania tego artykułu, zostałem zainspirowany niewiedzą, jaka, jak sądzę, panuje wśród posiadaczy komputerów na temat gier typu role-playing. Nie chcę wyjaśniać jak krok po kroku rozwiązywać zadania znajdujące się w programie, chciałbym wprowadzić jedynie w atmosferę i wyjaśnić problemy związane z rozpoczęciem gry.

Dawno, dawno temu, w odległej krainie zwanej Krynnem, gdzie smoki przesłaniają słońce w południe, a gnomy kopią tunele pod twoimi stopami, wydarzyła się pewna historia... W ten oto sposób można rozpocząć opis wspaniałej gry fantasy pod tytułem *Champion of Krynn*. Co prawda Champion... należy już do przeszłości (firma *Strategic Simulations Inc* na rynek europejski wypuściła ją w 1990 r.), lecz gra ta należy do typu gier tzw: "nieśmiertelnych".

Program powstał na bazie sławnych w latach 60-tych w Stanach Zjednoczonych gier planszowych Advanced Dungeons & Dragons. Bogato rozbudowana i błyskotliwie ułożona fabuła, wspaniale wykonane wstawki graficzne, nastrojowa choć prosta muzyka powoduje, iż złakniony nowych wrażeń gracz z pewnością zasiądzie do komputera na wiele godzin, dni... może miesięcy. Cała gra jest trylogią, pierwsza część nosi nazwę Champion of Krynn, druga Death Knight of Krynn, trzecia natomiast Dark Queen of Krynn. Rozpoczynając grę w Championie musimy uwzględnić możliwość kontynuacji przygód w kolejnych odcinkach.

Ogólny szkic gry jest taki sam lecz sposób wykonania, szybkość, a przede wszystkim skuteczność należy do gracza. Pierwszą czynnością, którą należy wykonać po rozpoczęciu gry jest stworzenie team'u, grupy, czy też jak kto woli, drużyny. Podstawowo grupa liczy sześciu członków. W trakcie gry liczba ta może się zmieniać. W skład mogą wchodzić przedstawiciele następujących profesji:

1) KNIGHT, PALADYN (Rycerz)- wyśmienity wojownik, stawiający honor ponad wszystko. Każdy rycerz wraz ze zdobywaniem doświadczenia może posiąść umiejętności klerykalne. W zależności od kierunku kształcenia rycerz będzie albo lepszym wojownikiem albo lepszym klerykiem: Knight of the Crown (lepszy wojownik), Knight of the Sword (taki sam wojownik jak kleryk), Knight of the Rose (lepszy kleryk).

2) FIGHTER (Wojownik) - doskonały żołnierz, szybki i skuteczny. Jednakowo posługuje się mieczem i łukiem. Wojownik, drugi po rycerzu, tworzy przednią straż składu.

3) THIEF (Złodziej) - niezastąpiony członek każdej drużyny. Tylko on potrafi skutecznie unieszkodliwiać pułapki, otworzyć zamknięte drzwi, odnaleźć skarby. Jest on słaby w walce lecz jeśli już do niej dojdzie dużym plusem jest jego wysoka zręczność (dexterity).

4) MAG (Czarodziej, Mag) - Czary ofensywne to jego specjalność. Tam gdzie trzeba kogoś spalić lub uśpić na odległość, zamienić osobowość, zasypać gradem lodowych igieł - jest on niezastąpiony. Gra bez niego jest bardzo trudna.

5) CLERIC (Uzdrowiciel) - jesteś ranny, chory, zatruty-wyleczy cię kleryk. Tam gdzie trzeba walczyć, niestety kleryk się do tego nie nadaje. Lecz prawdziwa grupa nie obędzie się bez niego, gdyż tylko z nim możliwa jest opcja "fix" (leczenie), pozwalająca na bardzo szybkie zregenerowanie utraconych sił.

6) RANGER (Strzelec) - dobrze walczy, mając dużo doświadczenia potrafi rzucać czary charakterystyczne dla druidów, lecz przede wszystkim doskonale posługuje się łukiem. Potrzebny w drużynie, lecz nie jest niezbędny.

Ten uproszczony opis zapoznaje nas z postaciami mogącymi tworzyć naszą grupę. Ważną rzeczą jest to, iż członkowie zespołu niekoniecznie muszą kształcić się tylko w jednej specjalności. Dozwolony jest cały szereg kombinacji. I tak na przykład mag może być jednocześnie wojownikiem, wojownik - złodziejem, złodziej natomiast magiem.

Nie pozostaje to niestety bez wpływu na postać. Osoba kształcąca się w dwóch profesjach, uczy się dwa razy wolniej od, na przykład, "czystego" maga, analogicznie w trzech profesjach, trzy razy wolniej. Na początku jest to kłopotliwe ("czysty" mag ma czary 4-ego stopnia, wojownik - mag 2-ego stopnia), lecz dopiero później możemy docenić możliwość tych kombinacji gdy np: postać rzuca czar na wroga, a nastepnie spokojnie bierze miecz i idzie go ciąć konwencjonalnie (prawdziwy mag miecza by nawet nie podniósł).

Osobiście bardzo polecam złożenie kleryk-ranger. Po pierwsze: umie się bić, po drugie: rzuca czary kleryków, po trzecie: później rzuca również czary druidów. Moim zdaniem, taki przykładowy team, powinien się składać z rycerza, wojownika, złodzieja-wojow-

SANERO STATUS OKAY
MALE 350 YEARS HIT POINTS 70/70
NEUTRAL GOOD SILVANESTI ELF
SHINARE CLERIC/RANGER
LEVEL 7/7 EXPERIENCE 150254
STR 18(75)
INT 18
WIS 18
DEX 19
CON 18
CHA 18
ARMOR CLASS -2 ENCUMBRANCE 1318
THACO 10 MOVEMENT 9
DAMAGE 1D10+5
TWO-HANDED SWORD +2
PLATE MAIL +1
ITEMS SPELLS EXIT

SANERO
HITPOINTS 62
AC -2
TWO-HANDED SWORD +2
MOVE VIEW AIM USE DONE

nika, wojownika-maga, wojownika-maga-kleryka, kleryka-rangera. Obecność dwóch kleryków jest ważna, gdyż po walce czesto zdarza się, że jeden z nich ulega jakiemuś "wypadkowi" wtedy funkcje jego przejmuje drugi.

Pierwszym problemem dla nas będzie listwa informacyjna każdej postaci. Podstawowymi cechami charakteryzującymi postać są:

str-strength - siła: Jest to jedna z najważniejszych cech postaci. Od niej zależy siła ataku a co się z tym wiąże ilość ran zadana wrogowi. Siła jest charakterystyczna dla każdej postaci, inną siłę ma elf, inną człowiek, jeszcze inną gnom. Również to, czy postać jest czarodziejem czy wojownikiem ma wpływ na maksymalną wielkość tego wskaźnika.

int-inteligence - inteligencja: W zależności od jej poziomu nasz bohater będzie zdobywał doświadczenie albo wolno, albo szybko. Postać mająca niski wskaźnik inteligencji będzie się uczyła bardzo wolno.

wis-wisdom - wiedza: Jest to druga po inteligencji cecha charakteryzująca szybkość przyswajania wiedzy. Wysoki wisdom najczęściej posiadają magowie i klerycy.

dex-dexterity - zręczność: Ten wskaźnik jest bardzo ważny. Od niego zależy skuteczność ataku (siła ataku), osoba mająca wysokie dexterity jednocześnie ma lepszy wskaźnik AC czyli Armour Class (klasa zbroi), głównym czynnikiem warunkującym wybór złodzieja jest właśnie wysoka zręczność. Na przykład człowiek złodziejem dobrym nigdy nie będzie gdyż jego maksymalna zręczność wynosi 18, a już elf jest o wiele lepszy gdyż posiada dexterity równą 19. W zależności od wysokości tego wskaźnika złodziej albo usunie więcej pułapek, otworzy więcej drzwi, albo nie.

con-constitution - kondycja: Często, gdy postać zginie, można ją uratować rzucając czar "raise dead" (ożywienie zmarłego) albo "resurection" (zmartwychwstanie). O ile czar zmartwychwstania nie pozostawia po sobie żadnych widocznych skutków, to raise dead przeciwnie. Po każdym, sukcesem zakończonym, leczeniu wskaźnik constitution zmniejsza się o jeden (czar ten nie działa na postacie będące elfami). Dodatkowo, w zależności od wysokości constitution różny jest czas leczenia zwykłych ran na postoju (im niższy, tym dana osoba leczy się wolniej).

cha-charisma - charyzma: Wysokość tego wskaźnika charakteryzuje przywiązanie danej osoby do reszty grupy. Osoba posiadająca wysoką charyzmę będzie rzadziej ulegała czarom wroga np: "hold person", "charm person"...

Te sześć podstawowych cech określających postać, to nie są jedyne informacje znajdujące się na listwie charakterystyki postaci. Możemy tam znaleźć jeszcze inne wiadomości:

male, female: mężczyzna, kobieta: Dane określające płeć postaci i nie mające większego wpływu na jej możliwości.

wiek: W zależności od rasy górna granica życia naszego bohatera może się wahać od 30 do 300, 500 lat. Wiek można zmienić za pomocą czaru. Istnieje czar odmłodzenia i przyspieszenia. Odmłodzenie daje więcej siły, natomiast przyspieszenie daje więcej szybkości, lecz jednocześnie przyspiesza proces starzenia.

psychika - evil, good, neutral: Trzy podstawowe cechy określające czy nasza postać ma charakter zły, dobry czy neutralny. Jednocześnie dana osoba może być w swej np. dobroci stała (laful), zmienna (neutral), i słaba (chaotic). W zależności od psychiki, czary złych czarodziei będą rzadziej działały na osoby z twojej grupy, bedące zatwardziałymi niegodziwcami. Wydaje się, że wygodnie byłoby stworzyć zespół składający się wyłącznie ze złych postaci, lecz tak nie jest. Po pierwsze ani rycerz ani paladyn nie może być zły, a po drugie nawet gdy wybierzemy całość złą (oprócz rycerza), to na pewno nie wpłynie to na poprawę stosunków pomiędzy dwoma w/w grupami.

profesja: W tym miejscu możemy się dowiedzieć jakiego zawodu jest nasza postać. Czy jest to mag czy na przykład ranger. Warto tutaj zwrócić uwagę, że mag może być magiem białym (white), czerwonym (red), czy też czarnym (black). Różnica polega tylko na tym, iż np: w przeciwieństwie do białego maga czerwony posiada takie czary jak: haste (przyspieszenie), czy invisibility 10'rad (niewidzialność 10-tego stopnia). Również wydajność czarów maga jest zależna od słońca. Czary białego maga będą mocniejsze gdy świeci białe słońce (górny fragment ekranu podczas gry - tam znajdują się trzy słońca) i analogicznie czary maga czerwonego wzmocnią się gdy świeci czerwone słońce i maga czarnego gdy świeci słońce granatowe. Sytuacja u kleryków jest również ważna, istotne jest w jakiego boga wierzy dany kleryk np: jeżeli jest to bóg "paladine", klerykowi najlepiej wychodzi czar turn undead czyli ponowne zniszczenie ożywionych zwłok. Kleryk wierzący w danego boga posiada umiejętność skuteczniejszego rzucania pewnej grupy czarów od kleryka wierzącego w innego boga.

status: Wskaźnik ten informuje czy dana osoba jest aktualnie zdrowa, czy zatruta, niezdolna do walki, zamieniona w kamień itd.

hit: Tu widać ile jeszcze hp (hit

point - punktów trafień) może otrzymać nasza postać zanim zginie. Na przykład: 95\106 oznacza, że ogólna suma punków życia postaci wynosi 106, z czego zostało mu tych punktów 95.

rasa: Czy jesteś krasnoludem, człowiekiem, elfem czy półelfem w tym miejscu o tym się dowiesz.

experience: Doświadczenie, tu widnieje ilość posiadanych przez nas ep czyli experience point. Doświadczenie zdobywamy podczas walki, rzucania czarów i sprawnego omijania zastawionych na nas pułapek.

armour class: Klasa zbroi, im niższa tym lepsza. Bohater posiadający zbroję o AC równym -10 (minus dziesięć) ma od każdego ataku na siebie odejmowane 10 hp (hit points), poza tym im mniejsza jest wartość zbroi tym mniejsza jest możliwość trafienia przez wroga.

thaco: Trafialność. Im niższa jest ta wartość tym większa jest szansa, że twoja postać trafi wroga. Osoba mająca thaco mniejsze od zera jest już bardzo dobrze wyćwiczona.

damage: Zniszczenie. Ta wartość ukazuje jak wielką stratę może zadać twoja postać wrogowi. Na przykład, na początku, przy wyborze drużyny, dana osoba ma damage 1D6+1 co oznacza, że w momencie trafienia wroga, komputer wylosuje liczbę z zakresu 6 (1,2,3,4,5,6), a następnie doda do tego 1. Maksymalnie więc siła przy ataku może wynosić 7, minimalnie zaś 2. Wraz ze wzrostem siły jak i zręczności, ta wartość może się zmieniać na plus tj. wartość damage przy zwiększonej sile lub zręczności może wynosić już 1D6+4, w momencie gdy dana postać ma w dłoni długi miecz damage może wynosić 1D6+8 (losowana jest liczba z zakresu 6 i dodawana jest do tego wartość 8).

I to by było na tyle... Mam nadzieje, że te informacje okażą się pomocne w dalszej grze.

Marcin "Curl" Kasprzak

CORE WARS 64

W natłoku różnych codziennych zajęć nie zawsze można znaleźć wystarczająco dużo czasu na napisanie czegoś dobrego dla KEBABA. Od pewnego czasu pracuję nad pewnym pomysłem (efekty tej pracy powinno być widać już w następnym numerze tego miesięcznika), co nie zostawia mi dużo czasu na tworzenie małych, zręcznych programików. Aby jednak czekanie na efekty moich działań zbytnio się Wam nie dłużyło, postanowiłem przeszperać swoje dyskietki w poszukiwaniu czegoś akuratnego do opublikowania w KEBABIE.

O opisywanym tu programie zdołałem już prawie zapomnieć, ale gdy go tylko wyłowiłem, od razu byłem pewien, że jego tematyka w jakiś sposób koresponduje z tematem mojej aktualnej pracy (tej od "efektów za miesiąc"). Jest to bowiem mój własny MARS (Memory Array Redcode Simulator)! Co to jest? Spodziewam się, że tylko niektórzy z Czytelników wiedzą, o co tu chodzi. A oczywiście chodzi o dość znaną swojego czasu grę dla programistów - Core Wars (czyli po polsku - Wojny Rdzeniowe). Co było w tej grze tak ciekawego, że swojego czasu (a może i do tej pory) rozgrywano w niej zawody?

Dla ciekawych spieszę z wyjaśnieniami dotyczącymi zasad tej gry: na pewno nie raz nasłuchaliście się fabularyzowanych opowieści o inteligentnie mnożących się wirusach komputerowych, które w sprytny sposób omijały zabezpieczenia i atakowały znajdujące się w pamięci komputera dane i programy. Tu zasada jest zbliżona - do pamięci komputera "wpuszcza" się dwa specjalnie napisane programy (najbardziej obrazowo można określić je mianem wirusów, a jeszcze lepiej - wojowników), które pracując jednocześnie dążą do zniszczenia przeciwnika.

Walczące programy piszę się w specjalnie do tego celu zdefinowanym języku, który nazwano RedCode (od Reduced Code, czyli w wolnym tłumaczeniu - język o zredukowanej ilości komend). Kod źródłowy tak opisanego "wojownika" jest następnie kompilowany do kodu wynikowego, w którego formie program jest umieszczany w losowo wybranym miejscu pamięci razem ze swoim przeciwnikiem.

Wszystko, co teraz należy zrobić to uruchomić opcję walki i czekać na wyniki, pilnie obserwując komunikaty z linii frontu.

Dla wyjaśnienia wszystkich pozostałych niejasności - zasady gry:

1. Gra jest rozgrywana w pamięci o wielkości 8000 bajtów. Obszar ten jest zapętlony, tzn. po "przejściu" adresu 7999 zamiast komórki o adresie 8000 zaczynamy od komórki o adresie 0 (zerowej). Ten sam mechanizm działa także w odwrotnym kierunku. Aby dodatkowo utrudnić grę, komórki o adresach 3999 i 4000 są tożsame swoją zawartością, tzn. wpisując cokolwiek do adresu 3999 wpisujemy to także automatycznie do 4000 i na odwrót. W momencie startu cały obszar gry jest wypełniany zerami.

2. W każdej kolejce działań (cyklów) program nadzorujący walkę (MARS) wykonuje po jednym rozkazie z programów "wojowników".

3. "Wojownik" umiera przy próbie wykonania rozkazu o kodzie 0 (zero).

4. Każdy "bajt" z obszaru gry składa się z trzech części:

```
[kod rozkazu] [argument X]
[argument Y]
```

5. Argumentami X i Y mogą być etykiety lub wyrażenia liczbowe w zakresie od -7999 do 7999.

6. Na język RedCode składają się definicje 10 rozkazów:



`DAT XX (kod 0)` - to definicja komórki zawierającej daną XX w polu Y i 0 (zero) w polach kodu rozkazu i X. W momencie startu cały obszar gry to zbiór rozkazów `DAT 0`. To próba wykonania tej właśnie komendy kończy "życie" walczącego programu.

`MOV X,Y (kod 1)` - przesyła zawartość X do Y

`ADD X,Y (kod 2)` - dodaje do siebie X i Y i wynik wstawia do Y

`SUB X,Y kod 3)` - odejmuje Y od B i wynik wstawia do Y

`JMP X (kod 4)` - skacze bezwarunkowo do X

`JMZ X,Y (kod 5)` - skacze do X jeśli Y jest równe zeru

`JMG X,Y (kod 6)` - skacze do X jeśli Y jest większe od zera

`DJZ X,Y (kod 7)` - zmniejsza Y o jeden i o ile Y w ten sposób osiągnie zero to skacze do X

`CMP X,Y (kod 8)` - porównuje X z Y i o ile sa one różne, to przeskakuje następną w kolejności komendę

`SPL X (kod 9)` - powoduje wygenerowanie kolejnego wojownika a dokładniej - uruchomienie kolejnego, niezależnego zadania, począwszy od adresu wskazanego przez X. Jak wspomniałem wcześniej, podczas każdego cyklu walki wykonywane są po jednym rozkazy każdego z wojowników. Po rozkazie SPL (od ang. split - rozdzielenie) komputer definiuje jakby nową "jaźń" dla nowego wojownika.

W praktyce przeprowadza się to w następujący sposób: kopiuje się program wojownika w inne miejsce i rozkazem SPL uruchamia się go od początku. W ten sposób mamy już dwóch "wojowników" po naszej stronie. Należy jednak pamiętać, że podczas każdego cyklu gry wykonywany jest jeden cykl z programu "wojownika-oryginału" a w następnym cyklu - z "wojownika-kopii".

Przy programowaniu dopuszczalne jest używanie etykiet o długości do 8 znaków (znak pierwszy musi być literą).

Dopuszczalne są trzy sposoby adresowania w rozkazach RedCode:

1. Natychmiastowy np.

`MOV #10,adres`
- wstawia do komórki "adres" wartość 10

`MOV #-20,adres`
- też wstawia, ale -20

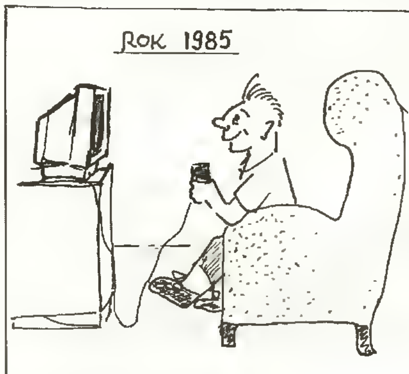
2. Bezpośredni np.

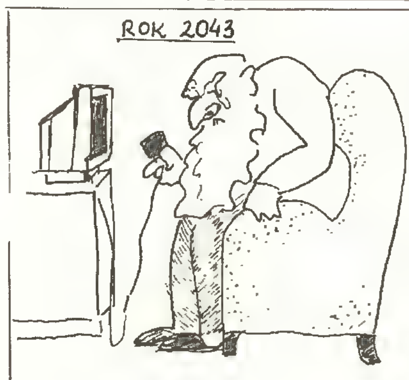
`JMP adres`
- skacze do etykiety "adres"

`MOV -4,5`
- pobiera wartość z komórki położonej 4 "bajty" wcześniej niż rozkaz MOV i wstawia do komórki położonej o 5 bajtów dalej

3. Pośredni np.
`MOV @-4,adres`
- pobiera adres z komórki znajdującej się 4 "bajty" wcześniej niż `MOV`, następnie w/g tego adresu pobiera z pamięci interesującą nas wartość i wstawia ją pod adres oznaczony etykietą "adres".

Należy zwrócić uwagę na fakt, że rozkazy `ADD`, `SUB` i `DJZ` wystawiają wyniki swoich działań do argumentu Y wskazanej komórki NIE NARUSZAJĄC dwóch pierwszych części "bajtu": kodu rozkazu i pola argumentu X. Ta zasada nie odnosi się do rozkazu MOV - przenoszenie danych dotyczy bowiem wszystkich 3 elementów wskazanej komórki z wyjątkiem trybu "MOV #", gdyż jego wykonanie powoduje wstawienie zera do pola kodu rozkazu i do pola argumentu X pod wskazanym adresem. Mowiąc krócej - tworzy się w ten sposób instrukcja DAT.





Teraz jeszcze parę informacji o użytecznych klawiszach w edytorze:

F-1 - przejście do Menu

CTRL + "strzałka w górę" - kasuje całą linię, w której znajduje się kursor;

CTRL + "N" - wstawia nową, pustą linię w miejscu ustawienia kursora

"strzałka w lewo" - powoduje wyświetlenie znaku podkreślenia, wygodnego przy zastępowaniu znaku spacji, który nie może być użyty przy używaniu etykiet.

Tekst źródłowy wojownika na dysku zapisywany jest z oznaczeniem "S." przed nazwą a wynik jego kompilacji - z "F." przed nazwą, np. "S.GEM-CANNON" i "F.GEM-CANNON".

Po skompilowaniu "wojownika" i jego wgraniu (tego z "F." w nazwie) do pamięci na początku gry, jest on uruchamiany od jego pierwszej instrukcji. Jeżeli z jakichś względów było by to dla nas niewygodne, instrukcj `@-x` można ustalić ilość linii, które przy uruchamianiu "wojownika" będą opuszczone. Np. zapis:

`@-2` oznacza opuszczenie dwóch pierwszych linii przy uruchamianiu programu.



Program Core Wars należy wpisać do komputera przy użyciu programu Code Inputer i przed uruchomieniem nagrać go na dysk. Program pracuje niestety tylko ze stacją dysków. Obok wydruku programu, na ostatnich stronach KEBABA Czytelnicy znajdą również kilka opisanych przykładów "wojowników".

Miłej zabawy życzy Autor wersji dla Commodore 64:

Paweł "Polonus" Sołtysiński

AQUATIC GAMES

Pewnego ranka, po powrocie z balangi u kolesia, po skonsumowaniu śniadania i wypiciu 2 litrów mleka zapałałem szczerą chęcią pogrania sobie w jakąś odlotową gierkę. Czym prędzej zaszperałem w 11 pudełku z dyskami i bez namysłu wziąłem pierwszy lepszy dysk. Leżał on w przegródce SUPER NOWOŚCI. Co prawda nie przypominałem sobie, żebym go tam umieścił z własnej, nieprzymuszonej woli (widocznie wtedy też wróciłem z jakiejś imprezy). Na tym dysku widniał napis: **Aquatic Games** - sportowa, dla dzieci (a więc na moje samopoczucie jak ulał).

Aquatic Games zawiera osiem konkurecji, postaramy się je teraz wyszczególnić w kolejności w jakiej one są ułożone:

100 METRE SPLASH - bieg na sto metrów z udziałem dobrze znanego wszystkim Jamesa Ponda. Polega on oczywiście na szybkim machaniu przyrządem drążkowym lub myszką podłączoną do drugiego portu (daje to znakomite efekty) . Można także przeskoczyć flaminga, który następnie odwdzięczy się nam podrzuceniem o kawałek. Czasy poniżej czternastu sekund można już uznać za dobre, jednak przekroczenie dwunastu sekund jest możliwe po paru ostrych treningach (najlepiej w siłowni).

KIPPER WATCHING - konkurencja ta polega na ochranianiu śpiących foczek przed piłkami zakłócającymi im sen. Nie można pozwolić na to, aby więcej niż dwie foczki uciekły poza ekran, gdyż spowoduje to panikę wśród pozostałych. Nie można także dopuścić do tego, aby budzik pojawiający się co pewien czas na gałęzi drzewka wyrwał naszych śpiących przyjaciół z sielankowej drzemki na łonie natury. W celu zakwalifikowania się należy bronić foczek co najmniej przez dwie minuty.

HOP, SKIP AND JUMP - skok w dal polegający na rozpędzeniu się, odbiciu, paru ruchach skakanką i dalekim locie po torze parabolicznym. Następnie lądujemy (miękko) na glebie. Niedaleko widzimy (po dobrym skoku) paru Latynosów w potężnych sombrerrach. Pomiar odległości skoku dokonywany jest przez starszego mierniczego pingwina PIK-POKA. Do zdobycia brązowego medalu wystarcza nam wynik powyżej 800m. Wierzymy, że po paru treningach (100,500,1024) uda wam się uzyskać spokojnie 1000m i złoto, o którym marzycie.

THE BOUNCY CASTLE - akrobacje dowolne. Naszym błędnym oczom ukazują się dwa tapczany, od których można się odbijać. Po wykonaniu paru podskoków każdy uzna, że czas zrobić nasze pierwsze salto. Po zrobieniu obrotu wokół własnej osi i salta na ekranie ukazuje się magiczne pudełko. Odbicia od tego pudełka pozwolą nam na wysokie loty i zbieranie chorągiewek. W ciagu 4 minut należy uzyskać najwyższe oceny za styl.

FEEDING TIME - mamy za zadanie uratowanie rybek (smoczków. potworków) przed wstrętnymi wędkarzami.Czynimy to przez ich regularne dokarmianie jedzonkiem z dozowników znajdujących się na boku. Dokarmianie stworków przez co najmniej dwie minuty pozwoli nam na uzyskanie awansu do następnej konkurencji.

SHELL SHOTTING - ta konkurencja należy chyba do najprostszych i ukończenie jej nie powinno sprawić nam kłopotu (nawet za pierwszym razem). Nasze zadanie polega na rozbiciu pięciu baniek. Można to uczynić za pomocą meduz łapanych do miski lub skacząc w kierunku bańki. Na skończenie konkurencji są przeznaczone cztery minuty.

TOUR DE GRASS - delfin jadący na jednokołowym rowerku przez wertepy to nie lada sensacja dla wszystkich graczy. Oprócz ciągłego kręcenia joy'em naciskamy strzał, aby przeskoczyć kraba lub złapać ptaszka. Wynik poniżej 40 sekund zadowala już nasze wymagające chipy.

LEAP FROG - bieg przez płotki i kałuże (wiadomo na czym taki bieg polega). Urozmaiceniem tego biegu jest to że płotki są pod napięciem (750 kV) i pomyłka powoduje podskok naszej żabki. Wyniki poniżej 28 sekund kwalifikują nas do medalu.

Gra ta jest szczególnie godna polecenia dla ludzi chcących się oderwać od szarej rzeczywistości (wspaniale dobrane kolory), oraz dla tych którym znudziła się Cywilizacja lub jakaś inna MEGAloMANIA.

W grze mamy możliwość udziału od jednego do czterech graczy, oraz trzy rodzaje treningu. Wydała ją w 1992 roku firma *VECTORDEAN* i chyba należy ją traktować jako miłą odskocznię od (przeważnie) nudnych i jednostajnych gier sportowych.

Przykłady procedur dla Wojen Rdzeniowych (C-64):

Zgodnie z obietnicą zawartą w artykule - możecie znaleźć tu kilka przykładów na "dobry początek" zabawy z Wojnami Rdzeniowymi. Przykłady można przepisywać "żywcem", no może tylko pomijając komentarze (mogą się nie zmieścić na ekranie edycji). Jak widać na listingach, komentarze za rozkazami umieszczać należy po znaku średnika. Każdą pustą linię (całoliniowy komentarz) należy rozpocząć od znaku gwiazdki (symbol mnożenia). A teraz przejdźmy do przykładów konkretnych rozwiązań:

```
[przykład 1]
*            "DWARF" (KARZEŁ) TO PROSTY
*           WOJOWNIK, STRZELAJĄCY ZERAMI
*           PO OBSZARZE GRY. CZYNI TO CO
*           PIĄTY ADRES, CO PRZY JEGO DŁUGOŚCI
*           (4 ROZKAZY) NIE POWODUJE JEGO
*           SAMOUSZKODZENIA
*___ _ __________
```

```
*
@-              ;USTAWIA ZNACZNIK POCZĄTKU PROGRAMU
*               ;ZA PIERWSZĄ LINIĄ PROGRAMU (PO "DAT 3")
*
CEL          DAT 3        ;ADRES PIERWSZEGO CELU
*
LOOP         MOV #0,@CEL ;WYSTRZEL ZERO W CEL
             ADD #5,CEL  ;ZWIĘKSZ CEL O 5
             JMP LOOP    ;STRZELAJ PONOWNIE
*___ _ ___________
```

[przykład 2]

```
*
*               "SMART WORMER" (SPRYTNY ROBAK)
*               PRODUKUJE "ROBAKI" CO KAŻDE SETNE
*               WYWOŁANIE PROCEDURY GŁÓWNEJ
*
*___ _ ___________
@-1             ;OPUŚĆ 1 LINIĘ NA STARCIE
*
COUNTER      DAT 100  ;LICZNIK WYWOŁAŃ
* LOOP       DJZ DOTASK,COUNTER  ;ODEJMUJE 1 I SKACZE DO "DOTASK" JEŻLI RÓWNE 0
             JMP -1              ;NIE ZERO - TESTUJ PONOWNIE
*
DOTASK       SPL WORMER          ;URUCHOMIENIE NOWEGO ZADANIA
             MOV #100,COUNTER ;PONOWNE ZAŁADOWANIE LICZNIKA
             JMP LOOP         ;...I SKOK DO PĘTLI GŁÓWNEJ
*
WORMER       MOV 0,1     ;A TU JEST NASZ MAŁY ROBACZEK
*___ _ ___________
```

[przykład 3]

```
*
*               "JUMPER" (SKOCZEK) - PRODUKUJE
*               KOPIE SAMEGO SIEBIE ODDALONĄ O 99 BAJTÓW
*               A NASTĘPNIE PRZENOSI PRACĘ PROGRAMU NA
*               POCZĄTEK TEJ KOPII
*___ _ ___________
@-2             ;"PRZESKOCZ" 2 PIERWSZE LINIE PROGRAMU
*
LNG          DAT 0    ;DŁUGOŚĆ PROCEDURY
TARGET       DAT 99   ;WSKAZANIE OBSZARU DLA KOPII
*
LOOP         MOV @LNG,@TARGET ;TRANSFER
             ADD #1,LNG       ;ZWIĘKSZ LICZNIK
             ADD #1,TARGET    ;ZWIĘKSZ ADRES CELU
             CMP LNG,#9       ;SKOPIOWANO JUŻ CAŁY PROGRAM?
             JMP CONT          ;TAK - IDŹ DO "CONT"
             JMP LOOP         ;JESZCZE NIE - PRACUJ DALEJ
*
CONT         MOV #0,91        ;UTWÓRZ LICZNIK DLA KOPII
             MOV #99,91       ;USTAW DLA NIEJ ADRES DO TWORZENIA KOLEJNEJ KOPII
             JMP 91           ;PRZENIEŚ PRACĘ PROGRAMU DO TEJ KOPII
*___ _ ___________
```

[przykład 4]

```
*
*               "CANNON" (ARMATKA) PRODUKUJE "SKOCZKA"
*               CO KAŻDE PIĘcDZIESIĄTE WYKONANIE PROCEDURY GŁÓWNEJ
*
*___ _ ___________
@-1
*
CNT          DAT 50   ;LICZNIK DLA CYKLI
*
PETLA        DJZ MOVEDIZ,CNT  ;SKOCZ DO "MOVEDIZ" GDY "CNT" OSIĄGNIE ZERO
             JMP PETLA        ;NIE BYŁO JESZCZE ZERA
MOVEDIZ      SPL LOOP         ;WYGENERUJ NOWE ZADANIE - "SKOCZKA"
             MOV #50,CNT   ;PRZEŁADUJ LICZNIK
             JMP PETLA     ;I POWTÓRZ CAŁOŚĆ JESZCZE RAZ
*
LNG          DAT 0    ;DŁUGOŚĆ PROCEDURY
TARGET       DAT 99   ;WSKAZANIE OBSZARU DLA KOPII
*
LOOP         MOV @LNG,@TARGET ;TRANSFER
             ADD #1,LNG       ;ZWIĘKSZ LICZNIK
             ADD #1,TARGET    ;ZWIĘKSZ ADRES CELU
             CMP LNG,#9       ;SKOPIOWANO JUŻ CAŁY PROGRAM?
             JMP CONT         ;TAK - IDŹ DO "CONT"
             JMP LOOP         ;JESZCZE NIE - PRACUJ DALEJ
*
CONT         MOV #0,91        ;UTWÓRZ LICZNIK DLA KOPII
             MOV #99,91       ;USTAW DLA NIEJ ADRES DO TWORZENIA KOLEJNEJ KOPII
             JMP 91           ;PRZENIEŚ PRACĘ PROGRAMU DO TEJ KOPII
*___ _ ___________
```

PROGRAM Szperacz;

```
CONST
   Szerokosc = 24;      {szerokosc macierzy w znakach}
   Wysokosc  = 3;       {wysokosc w wierszach}
   IloscSlow = 3;       {ilu slow chcemy szukac?}


   {w tej tablicy umiesc wlasny wzor macierzy}

   Tablica : ARRAY[1..Wysokosc] OF String =
             ('ztqifariliqicoklrqvxqkta',
              'praqmvrxvrhmfwrfmpvrmebb',
              'taedhqbzolddkebabredtoau');


   {a w tej slowa do wyszukania}

   Slowa   : ARRAY[1..IloscSlow] OF String =
             ('WFMH',
              'QRT',
              'Kebab');

VAR
   Wiersz, Kolumna, Nr, Indeks : Byte;
   Slowo : String;

PROCEDURE Zaznacz(Y, X : Byte; AddY, AddX : Integer);
BEGIN
   Indeks:=1;
   WHILE Indeks<=Length(Slowo) DO BEGIN
      Tablica[Y,X]:=UpCase(Tablica[Y,X]);
      Y:=Y+AddY;
      X:=X+AddX;
      Inc(Indeks);
   END;
END;

PROCEDURE Porownaj(Y, X : Byte; AddY, AddX : Integer);
BEGIN
   Indeks:=1;

   WHILE Indeks<Length(Slowo) DO BEGIN
      X:=X+AddX;
      Y:=Y+AddY;
      Inc(Indeks);
      IF (X<1) OR (X>Szerokosc) OR (Y<1) OR (Y>Wysokosc) THEN Exit;
      IF UpCase(Tablica[Y,X])<>UpCase(Slowo[Indeks]) THEN Exit;
   END;

   Writeln(Slowo);
   Zaznacz(Wiersz,Kolumna,AddY,AddX);
END;

PROCEDURE Sprawdz(Y, X : Byte);
BEGIN
   Porownaj(Y,X,0,1);
   Porownaj(Y,X,1,1);
   Porownaj(Y,X,1,0);
   Porownaj(Y,X,1,-1);
   Porownaj(Y,X,0,-1);
   Porownaj(Y,X,-1,-1);
   Porownaj(Y,X,-1,0);
   Porownaj(Y,X,-1,1);
END;

PROCEDURE Szukaj;
BEGIN
   FOR Wiersz:=1 TO Wysokosc DO
      FOR Kolumna:=1 TO Szerokosc DO BEGIN
         IF UpCase(Slowo[1])=UpCase(Tablica[Wiersz,Kolumna]) THEN
              Sprawdz(Wiersz,Kolumna);
      END;
END;

BEGIN
   FOR Nr:=1 TO IloscSlow DO BEGIN
      Slowo:=Slowa[Nr];
```

```
      Szukaj;
   END;

   Writeln;
   FOR Wiersz:=1 TO Wysokosc DO BEGIN
      FOR Kolumna:=1 TO Szerokosc DO BEGIN
         IF Tablica[Wiersz,Kolumna] IN ['A'..'Z'] THEN
            Write(#27,'[32m')
            ELSE
            Write(#27,'[0m');
      Write(Tablica[Wiersz,Kolumna],' ');
      END;
   Writeln;
   END;
END.
```

"CORE WARS " $0801-$1FC8

```
0801: 2Myg 1FU! c3gV 0a00 ue&1 Ki0v 98
0813: CvE0 !d3T j001 Qfle RYAY QeI0 16
0825: 7qzM %030 dZ3z x05& je&L tYtP 17
0837: &6!b DhYw YgzE dG90 G&y6 4J3X 6d
0849: Phbg Yc2q rL5c CH9% 1#Ay jsI2 28
085b: 4&Tp 24IB ak28 Lp3K 6qTg 2cC8 19
086d: Qe!e ZrwZ !fEg &bR0 let! &0Rg 69
087f: gqbV eM00 87Dx s1tK s1S7 kj0U e5
0891: 8Na& 9SNb kqht 81y# rk0U QDQw ec
08a3: epEr IMY7 4w4d 5vvp 6h&f 3myL a0
08b5: zMY6 npMG PNk1 bNC% xkhA IgYw d3
08c7: UPMY x&69 zG2d x&Wl EdKs 4M17 8d
08d9: pCAR 59yt rMAe 1!kT 21DZ Hsen e8
08eb: 1lV4 1Nvh 0MIY 0wYD 0w&1 3wck 2f
08fd: 2&td Iizj nMfQ jX72 d!1G CD3c b8
090f: cg&# 8IU! 3zjj 1x8M m5m& p3wr 5c
0921: 4USv Plu6 &!zH bfbB fwLY 4M4m a2
0933: Emg1 #Icd 40Ac gH%q 2v1D 3s40 57
0945: PjMc Z0Wh 4WGJ s0c6 50k& CFsf 4d
0957: 3wMp 8cIY 6N7r byQw 4hmD hwFy 9d
0969: H20k 25Ae Kx41 11Pk sYda PdxY 65
097b: 7X0Y Hr%j ggeJ M2Vx %18c ikXe 55
098d: h4aD jzF3 V5Re gA6P g#zz EQ5Q f1
099f: EG4m alTW QuMY 4If4 UHP0 XLsY 18
09b1: VljK Vuk! x#p7 SKZY WLcj xU0v aa
09c3: &##Z XKHR fei! 5Lp& kKYj XJQN 38
09d5: #jMu q1rx WwbQ f3Vp XKX# #4Cb 50
09e7: Yv%X etkY YkNQ %KUY XyDm UuXD 40
09f9: fbWK nCUu LzX# fEWu zyXu %EV# fa
0a0b: zz4x Yu7# rLYT 620x 8yfQ 9!wF 34
0a1d: ayII biU0 c34! cPgR dzsU eg0X d3
0a2f: 0DUb dP#0 #0fC Eg&7 20Aa 2MMd 27
0a41: 3wYg 4u!J 51km 5Nwp 6D9m g6g0 e5
0a53: U3!E !GGZ 00%j ivM9 84gr #Mgg 93
0a65: ECav 00Mh l!nU DU22 T0!J WYM! de
0a77: yFuv 093q MQt# 0zbf e2&r 112x 0e
0a89: D25U Igcg ZkPb 8q0I yg8b HgG5 06
0a9b: 2i2d gwdc m20! akPE 322m t8cK b1
0aad: wTxc VNV& GT8Z 6ES6 0Fmk 6nFp e5
0abf: j4T3 E1Jz 13U4 #gk4 2hgA 7pEw 95
0ad1: 0Tvc 1rFY 4CqC 5zcb 1g81 0ysV 08
0ae3: cU!2 AW%R 2EuK T&#i xdSg wpa1 7c
0af5: zqUw 82Qg 4wkj 4%4b ci06 3WjW f6
0b07: 3gke 5iSy jWA5 TwX& Liwg auM0 4d
0b19: YG8% a4sT zcE! 8Kyt g0fE %jPw 80
0b2b: H1TS nF31 Gg7x Kwjg zhKs 7c21 e5
0b3d: 5Md1 zLEe 3&TU 1WzE 4002 VNnh e4
0b4f: 63Nb Y0J1 2wEK 4d0Y a%md 01c& a7
0b61: bxjh WziQ #EQ1 DSMP 12SY mGW4 d2
0b73: GE9U Gn#d 3tP3 swXs W8Uq k&kr 52
0b85: qPz% zhbg Gjad 5a6m NU01 0Rxc 6b
0b97: &t&Y Y6lw aQNj UHM3 zjch EwaZ 5e
0ba9: mh6d 9Z3E UedH V0aD zA8h 8dp% b4
0bbb: rj7G 2g88 2wYa 209c CZrX 9K7q ac
0bcd: CeSi S0le r0&z 5qhd 1Tdx H1t6 ad
0bdf: MIdA N8wH 4k2p 70hf r0kD 1H&j bf
0bf1: Qfxw ybBg 1Wuy Cd9u &239 4eUp e9
0c03: Q62R 2wfa A0!F 97Aw RhaF Ya2i a5
0c15: 4C00 Yz!F 17!m lY7Z HCMY !IHM d8
0c27: 2F&K rgr2 NXU1 Zr2p X!16 4L3j ec
0c39: GykI Sw#f an#K Vdjw tNbD HVH3 f3
0c4b: PJSn 0bEh Q2Dk p1yM 1KUy 6EZw 77
0c5d: pz3U 0GsM 2AE3 Xzxl KCmh Q2Rp 99
0c6f: IIA3 Aai8 a&Ip 6cB0 Q0km uL34 cf
0c81: 08tM H#xh 0YX2 uw1c rl2c 3tMb 46
0c93: 7kNa gGcj Q1Qp E8hi CqB0 zhfb 34
0ca5: Ggad SumU 5jKW 7t0Q F&4D Yf8i b7
0cb7: GX0q A1c8 GgCd xPra 0gq% 8f3h 0b
0cc9: 4B7K js4c DL3K !pTg 6TeI xZFe 40
0cdb: E4c8 WYXV Btgc Yfrg Nahq LxXp d0
0ced: YbI0 sCJc 8&g7 !gD9 nW3% 3a1j af
0cff: 0uLZ YiWS v1dy jeqh HTHY Exuw 15
0d11: 1WAf I%58 CwSF 3df8 M2zg niGC b3
0d23: eeru j7xe 2MXg skZj 08jP &3Ea ed
0d35: 35nJ %rgd dpwx ryhG 5wiL %LyS 6c
0d47: 9yRr 0fTY ZJJ7 %iIM nyhB NskJ 02
0d59: SQPQ 4QMd paP4 Nvac PUsm I4qh 7a
0d6b: #Yyc 9yYc 71gi !hXg pi2U 8KBg 45
0d7d: tiQ0 k3WV 1fB5 Zf9Q 4CQS 5Ngw e2
0d8f: niem kM4V zS&2 jkPq dfDd He50 c7
0da1: U!h& 2jiB %wVm AdKB %sCM Adkw a0
0db3: p15c Lib9 xsUn Tm1n 8a4h u22z b9
0dc5: fN&l e94& VlyJ 4ds2 Uh41 zgsh b1
0dd7: Hsv3 Xfkg Hswg zvEg k6IR 00bA fb
0de9: zw0d u971 r%p# K3D1 00Qv 7dy# a7
0dfb: SqwF 3x%q iCan cSbg UUE9 MUs4 fd
0e0d: 3MEv 1jUb v0qZ 00N8 CNS3 Qe1c f3
0e1f: HybG Ivla x&3X hr6d 4cB3 Q0ph 58
0e31: JgzQ 0C01 000i 7iG1 ME6z j&ww 61
0e43: ZjMB N8Zl eNs1 4!07 NvzN 12XW 68
0e55: zwPV fcBj s6Ew V4!# 087r hDM3 0c
0e67: bYBb VhfM 7IBh Q8my 9WA1 xzLr 05
0e79: Lr3c RMn! R8F# !kUU VaW1 5vcc 79
0e8b: 4Fkb #C1% #feN 9MMc 80Mf 4!3r ca
0e9d: 4y04 0hg1 aa!d 6i0p 3Nkw 4Nki 0c
0eaf: 1iwp bMUF fWT4 7eJb L64t zuEl 70
0ec1: GrEq DYA2 w3P% #ylN ihZc IK4V 9b
0ed3: b%pc Fytk eitW rz!W ejy% ujCs 53
0ee5: %zUU GvQD %cQ0 TP!y 4r4q Kj74 6c
0ef7: HiyF kM&! Vy3l xKLn DWDw Ef8E 16
0f09: cuLk !h3g Yqmt cdn3 I19c PyGz 6b
0f1b: N!08 A4K7 D#ve vhLa 0fMY AW&E ce
0f2d: e!wA fkh9 kzBI wfsY 3BHA IdAN 49
0f3f: AmsM sD4w Rkyi 4W#M EUnz pNM4 20
0f51: &zgM 9I2e sjzF abaA NP6e fZ3v dd
0f63: Vj6e NhSF %V7X hYK5 Gq1d taEw f5
0f75: B1CF ztcB Juqw S3Zb waRP 2a3L 83
0f87: BjsQ By3C 9fKF g3a3 5CiK RSDy 35
0f99: GF!K I5y5 0x31 4p32 uduy CxqZ 2b
0fab: EMXp 6xJc H1sK 2rn8 WAP1 63FJ cf
0fbd: Mx3B 5M#r I2Gw 6bxk 4%mb jxfS 1f
0fcf: GwU& M!&Y Wj2p bY1w vl8c c2Tw 71
0fe1: 0sww zhBv 5h00 u%69 tVMv dgz5 8a
0ff3: 9GDg Rjkl 611m g0qa SeTC 0zAQ de
1005: !vva aqAN E1CL pNri rdhf j!1d 8f
1017: HFwY j4Be Fi7M EyR7 MKae !1nJ 23
1029: Cyxz zREp H%Lr Rw2K tak2 4qmf d3
103b: XL#f jbQn pCSn b62F 5DGK r&PB c6
104d: Gvyr 33x1 bWA! d6NU YUTh 8Qma f9
105f: %ZXg vZIC Hrcp !t30 SLjs zEQ0 4a
```

```
1071: M9#y ga1F fUnY GtyI hDx# EyzF f4
1083: 4UQ1 V46x Q0Z1 nduS #Yy4 G!Z5 40
1095: h1%1 wv3p !iHM Rss# 5kIB !c2D 8e
10a7: DLmF 6q1F !6fi UGni Nsj3 PYj5 dd
10b9: 84Zi h5DT iw3K MtSD E3xg &hhG 0d
10cb: 32Jz eg!V 5G&f NC0m &Pw& wM6d bb
10dd: Mh1x 2HgY g8YT 5uUL 0CMc h1Jx ff
10ef: Fx%# xhzZ FtJd Z4yx ZgtY 34Nz 6c
1101: 1b0N 2Jw& 6CMc k6cM W&&d Gj6& 89
1113: iVVk 7G09 R1fg 4k0q 46wa VXEc 83
1125: td3z 7Eav ma0D iRUH E02h GkuD 16
1137: Utew c9wD CDrW zs9b uFpW 5x#D 16
1149: 4KEw JHp1 pKVD Xwd8 V3cr WKGJ 79
115b: #gEJ tN!e eGBU Eaq4 PAXx VV!p db
116d: k0is sgaB PNNH A6Be 1kSm VR9# 7c
117f: jc%1 NXwh 1PmV 63zG 8cY& yd3R 95
1191: H3#i S!Z7 Un0p #LFr YQMa K3fz bf
11a3: gsUh R4BA thbL y30D Kk03 GL0x 2b
11b5: e6WE pgEP &YhE 4yhz SexS RNVr 91
11c7: zgZm YKqv qmUs 4Hsy Cvhp rvHZ 81
11d9: Scpr QvvU 6Mvf Kg&e 7qyy &%V3 91
11eb: AvRu Xd2B %s7f NvhL RND8 !a8G 12
11fd: 2D95 KH9j EMhr iWVd GB6I CFpw 65
120f: fae5 7e2c 7J&P 6IHu ywWt &Ift 47
1221: jdw1 Hgzl Q2uF Tq1M drlh n1Sj 3c
1233: RR9f 3b&z P452 hk!D YC1U UZdR f4
1245: VqUF ct4T &wYJ %%0b !jLM 1Rw8 12
1257: !5vH U03M Lely vtSZ cYCv nfzQ d9
1269: 2!2& 7vdw 8qR1 F0ga 9xz! Vi5Q ba
127b: AF0L AwuN wWse QNPp 09%M 1KVi fa
128d: qQME 7kMx WWX7 &C1E q4Mb 6X4& b6
129f: wFca up6C BhLk 5uBC PpUt zvY3 c5
12b1: HnVw 9eaf bdyF 3NHC CHgQ &Mpi 51
12c3: seNE hydq 8EsQ D5zS yRgy &BMD 39
12d5: 5vct YyGu cW!E i8Ni shPE j1kt ec
12e7: H381 thRz yhCJ P&la 13ku 1iNh 96
12f9: wXJ4 0ETs mKN0 KYgs j7dg E3zS bb
130b: GgcZ ihsA XGA6 QeGF 1Z3C LWng c7
131d: Uy0t 7f7W w0eA zNM0 Q0Sy H4q0 fa
132f: 2kzZ zIzg XsAJ Q0Jp &E3N gn8u c6
1341: x0cc ez2g 6&#k Qhkw 91KJ kgdA e2
1353: Lp2J k0d4 rz0w Px!Z 15nd Kyg& 80
1365: hQ!X 7K0k eeRY J4iJ &iB% zm56 fd
1377: 1x!J %EYU %g2f z&43 G&0d wwdc 58
1389: MX1L 18Rs 7AN& 7GA9 suIL p4Me ca
139b: 403Z f0!m NT7W %eaZ #YD% Y1iB 2e
13ad: #NxF 1gST sL!5 #WnY !t2g VQM5 00
13bf: 5&wg U!20 5Bdv 28GE hQt5 y9p4 c0
13d1: 1gvZ MRG5 %GDZ FLKA %23& %Q!% 8c
13e3: 54x8 j5sL JG%0 NGRg xt7p 9v0a 66
13f5: e3Ew #S7S HA!A Y0Ww 1upT 7raz ef
1407: yd3T juUI z0Gc 6h6h Nsjg #A&Z 14
1419: q1Dc be5h k2Pv &aXB Jv0t Ikng 4d
142b: 0Q61 PC1H 1h0a 4kCS GvV! Cwrx 87
143d: US1U A#WK 9VAN mkUw 6Gtd XCAE e8
144f: #R91 mi0F Zkh3 j#vl zNat jRax 12
1461: HM1v w2tj drPp a2gF iwU1 zhzg bb
1473: 3bRW PAL& &i0Q %89g T9gW 01E0 47
1485: TJmh cXdN YRxc wiyP 0HSF X0DY 64
1497: GiWd 8iFa hEQK hG82 UzS4 &gw6 b3
14a9: 0uzw 3J3N U0bg 1uHG j3Ex zL1Y a8
14bb: FWQy 0sAA HpcX 8cwm Hngl aheF 56
14cd: Ma2s D70e gItw c!jB %Yw# kex6 2e
14df: 7#Uf H&Mw !13g XAM6 8nzI RLC5 c6
14f1: R0Xs EDCD E5CW %!Gu 0q8U 0BCZ ba
1503: YiA0 D0ee wpWr EY%% Vw4J Ftz8 66
1515: M0mk hzqz Fp3M JmM2 M6X% xHEg 78
1527: E0Fm dq4Q 8i26 &yMN 6Ekm n0Pd 9c
1539: L!MH e12a wi7P NPpC TaB% !J3W 2b
154b: b47G Kg2J pU4F 3Qgz 62C6 !8Qm e7
155d: fa3F fyIg qq90 dawU #sEY ij#M 3b
156f: pytE evAx W4gD XxwD x2rS cLbR 94
1581: 9Dya Hw&8 Y2re Nh8g 1qCP ncHg 4c
1593: YWsk !h#g 4x1q 3SUZ 2SKW 0gC% fe
15a5: gAPu %Tx0 7DF1 NUcV T4#8 4&wP 24
15b7: Gige db64 8LJh jQIK hg&j KQV1 7c
15c9: jpbf Pzan ATPs C15v 87WK 7M6Z 39
15db: 806t wf3! JAgh I8Sp m4fy 89Qw 20
15ed: 8dML I4M0 qAx3 y5UA 8cQg 442J 5e
15ff: &qvC jjER rNXq a14n Y4Ss 0J#A 0c
1611: jc4y xIqy #FGy W1rS %1gN NnwU fa
1623: 601& &aDN E2bc nz3G ccUL 6wGF bd
1635: 23yI Hg50 Hw10 T4&# RC4t rMmu 0c
1647: Nq60 wVn3 k4Yx Dqdn eJHh 9W0z 16
1659: kQx0 8MQF sh!f mvZ4 hkPI xcx4 3f
166b: tpe! 4#83 ULkq 98ZI PuLZ eeT7 a8
167d: 4b02 NLW5 %1Qh !V4N EyaZ vjg0 83
168f: 15Jk XT4f 2!U5 3xg5 4y0c 2gQ9 a1
16a1: 520f 1y03 6gcc 1hcW 8!gM zvZq 92
16b3: cj0p Tg1Z 3Es9 w121 kaH9 3v0y f5
16c5: hq#g YYB7 Ie%9 gr1K BPGM V!qj 9f
16d7: fk3E yyA7 wgax 8WVh 0FkF vVQw a9
16e9: xq87 L1CI uwnB T%&M wcZN Eycw cd
16fb: Xyie 3yjj ygHY fbys cd1h I0me 2a
170d: 2wA# AYcM 3n44 zi8A 9C3m e1Qs 51
171f: V8Ji ZWQt 61XV Ex6e 7PMw YAki d5
1731: 2aQx 1aUy 123T 8UQj Mnti JUWd 50
1743: %F#d %#qd vWBI unhm j90A !kid 26
1755: U1BZ Nk&N kQx8 A94e bvJm xXcE ce
1767: miZe ajY0 mmI# D098 8#RE dj43 1d
1779: PMkc vx%1 1YCw 94UQ DsMH 11DS cc
178b: 6IEG 9&iS KQ5e &kJ5 mj2N !bR1 7f
179d: 7%K0 iBq6 11f8 Nit3 frb& h51i a8
17af: Q%d7 rRI& k%Xa 1r%w zCBS xQt5 9c
17c1: zMeQ &4P1 &yVD KPVr Ngkq 4Rhc 65
17d3: BD6d fm#r 61cC BYf0 FDh& XL3X 0b
17e5: u5Ay V#j5 sHrs XdU! xZRr AD#r 99
17f7: biR0 TBK2 YmS9 ai00 UFq4 XIAM 6d
1809: skTM 1BzP mSbR 9rwB Zynx BAMc 71
181b: 9y%B v52# Z!Sf MiAH a!Pc 5!SU c3
182d: LF7J tFVM 397B q5ye Ydeg #63S 39
183f: 5v0S g0cn TCqk XDwJ a!2I 3Z1r b6
1851: s1k7 xice aNEw t0c1 V5bd Fi11 97
1863: 1i4S IGky mb82 Vyek uPA1 NZJp 51
1875: 6Evp &EEw 1!Zp WEVN !tpU Qaqe 81
1887: yiyG LvAB zt4C Lg8C zt8C 82ZD 5a
1899: #yGY rHTF Reec 1JNQ D8Sj wVmg 96
18ab: zDn9 T!el Y!vK 5Qa5 3uUm agzK 3d
18bd: 5kMx XKky krQg 2dQk HXA6 !x3R 95
18cf: j2j6 &HEE 6uK5 hfJq j9ax f71m c7
18e1: d2ff DmP9 gMTT 9aiY G&0s 3Cmh 56
18f3: wVe5 ENgj cwKT E2un P92C egZ6 3c
1905: j7gx 08Kh 62L3 s5tf jxB8 UHIx e7
1917: wU4y 13sk fyt2 g1hk j5Cn gAZi be
1929: j1V4 vgQL njxn 9QUZ k#DA I0kr 08
193b: 8d0w hemF dmuP jekK k59f hR91 81
194d: ji1g jQBe !Bua i1cM 1nxm ezxc be
195f: ikR9 15kg 9R0& iB1L 3bkd 3hkN 6c
1971: ryxP 7Mzg Wv&E Nw6F Ea0E I&pU ed
1983: Vw5c 7yAq ba8m j4Zj 14sh dhyR 5b
1995: 1J8h hBFF NsQ9 U14V Pgzy 4gqF 8d
19a7: c8RH rBn1 EspI Bmbd 2MMm 4y%d 63
19b9: 2w7g 2G9e a8aF g8QC khgB %xw6 28
19cb: 14f# zndc e2p% iC8U Wg8! MTUr 1f
19dd: chHs VvQC j4EH nifA !NiF 8ZAx 4b
19ef: zwyE 0w45 2&1V gd10 FtQF jdMF fd
1a01: 1s1R I8E# %9CN iwt9 s!2H !fG! f1
1a13: bGxI 76y8 Y9Me iUuh ygD7 i3ca bf
1a25: 5ScN JA5J H48S hwhK &Q!V x0QG 2b
1a37: j9V7 mC4f 80DV jUM3 4&ZU ak3M e3
1a49: qZ91 f4fK i4iJ HVpb T5qg &r22 07
1a5b: fkMt 3It& Yjpz 0dBN wyE8 IET& 38
1a6d: DAad BzRx 1&jx 0106 j5sB NKMV 1a
1a7f: JALK 36s9 1S6F b&S! 0WX0 icea ad
1a91: A!71 2XW9 C2YE xxz% HvUL ahg8 39
1aa3: u%wG EbU9 QGVa LcAf 7cA1 PcCv 75
1ab5: Q1Ea X&2w 1q80 LqJ8 !ezw 1t3R 85
1ac7: &cCw QfLk J1mp mFeF xr7q GrP# ac
1ad9: vmGV 0fym XFzU JDNh W9g3 fvwI 6e
1aeb: sm0k &U0B d8i1 3EW7 QcAM 6kdR 06
1afd: U5jd 3H3Q 1G5N WKzh HgWv 1dQk ac
1b0f: tngg 6M4T PpvM 190a I12a T1ig 6f
1b21: %wMr 6aTz 4wVm 2gt5 iHhJ Ky3s c2
1b33: 3V7C UeY3 Hr4F weV% zs4k.FbuB 34
1b45: RWWs qiMw nCMT e5XG gaQ& BBYp e6
```

```
1b57: Z1uw gBcP Fu#z b71a AqLx 63Rh 28
1b69: cRBM uSfN c%3I nbTW E%Ih m211 c9
1b7b: %h3e c6rg 1&hi Yfjw xey4 8#1W 00
1b8d: 830g bz%g 38xj Q0u8 3gC0 saxf d4
1b9f: g#Fw 8vTy Vaiq c0fF 1smI au0d da
1bb1: YQ1c TMCz !1Aw 1!HN 18pa 8e41 a7
1bc3: Q0Bn acSM 0Z1E y#AI EGA2 zqpc 08
1bd5: n!Ph 97x3 AqSn CWHH Gf2d %DSE 8a
1be7: HvYL cF4q E7VI kaSx NDYF 7UQ6 ca
1bf9: 22wg 1y3b KnnT bi2s 8qU1 M&Sd 85
1c0b: fbi0 QxcC uS83 Gj4c Gjad c!VI 0d
1c1d: D%0g 5gji aq12 8PzH n7xc h!Ud 8f
1c2f: hAB7 i5hP Ci0z cy1d nhR5 %5ww fa
1c41: jAln 85h1 kQJL 4t!N 12X9 9t0H 77
1c53: S0Sk g9gi g92U g1hc 5g38 Zih0 d8
1c65: iI89 vGF1 qe0d gV79 kb3N iUmd 0a
1c77: kNxF 0F03 XwKf H5bG 45mZ bAPA cf
1c89: 3Nid L!Vw !b4w i2Cv 2i2d M2VE a2
1c9b: &0cw WIdZ wUQ5 zuGr Hw3K m698 ba
1cad: Qwg8 eaWJ 6wIw HhI8 xi5c JyWT d7
1cbf: hQ9u jfQD u54! 6X%z 82bA 8cg8 b0
1cd1: qEkg z&!J 240A zgWJ 2w5w Hq83 3b
1ce3: !ifg 9AP2 aGkw xiaB 8&kz Hrfs 1b
1cf5: XM48 Hrg3 zg88 84Av E0iN 8F4w 26
1d07: y13V jfIG 838F j1wL Exp8 G&xg bd
1d19: vwDg #YQi Qf3X !J3N q4NU LqAT 11
1d2b: xg6B &p&H Gnh1 0Q4L Gprk B!Zc f3
1d3d: ENgd jAYb Xj9j r6NM hkUm xkZ6 ed
1d4f: 84df j119 j45k ikZe bw2y k4N3 06
1d61: bM3Y fdTL lwiJ 0#km LW0L 81UR f4
1d73: Mk34 ELFU CANG JgRc jQ54 841i be
1d85: kAZi 8gQd k595 kRcw kR11 gQkK 7e
1d97: %vUJ !IB2 YcyJ 7Mxw muS5 Nx7f 1d
1da9: jf0K Gg2d %UVc 81vh 3g0T f03w ee
1dbb: f0vm EC2w E!2Z muM8 E00w KH4m d1
1dcd: Mf#y sYrH 0D0c Fp3g 8t01 Gyfd 8f
1ddf: qaAX G!3f t1fg #aAd 8db% Hg7s 96
1df1: !n%g RqA1 8ceN LIP% gaMw QfBw 12
1e03: 0fAY Ex#Z 81mt M0va 4fsw Vf%9 2d
1e15: jL04 47#3 Qfn9 mv0b EyuF 8cz8 8c
1e27: !aA0 xh2C #P0d 1LEG 9x36 #Uzg 2b
1e39: YGE& &4yx %UnW Ewu6 ##j% FLXg 6d
1e4b: 0Ir% NLVE AdTw VZ3p Gju5 01xc ee
1e5d: 2MyF 4e&h 9165 1aGY S0sw 90uC c9
1e6f: 17SW 1QyB 47T7 1WxE Fwbg 2c00 ca
1e81: Q0j9 0f3& 66nY GFxB %&19 F17M cd
1e93: 88EU Vh6M 0Yp9 e818 FvPB 4r02 2b
1ea5: NLS5 %b58 y97Y Nw7K 8d3C 0pzg 74
1eb7: Ym0E 1MMe ag8a 6zFX %7NY 010w a1
1ec9: g800 0000 0001 0w00 0000 0gc7 05
1edb: 3Mc4 1g&7 1Mw8 10g5 1ws9 2wIc 29
1eed: 2Mzf 1VU! c3nr LG00 ubW2 1Vq1 7a
1eff: y33U j2g0 Fi6v b!b6 11hy aS2N a0
1f11: 8p42 %SMh !FTU bZ02 NyX8 Kgry 1f
1f23: a!gu Ki0u ChM7 !d3T 82c7 Y4&w c2
1f35: 8Mvg c20y 1SA2 !gig 9Z07 82c7 07
1f47: qgjg 7y0x 1SA6 !gTg 4sww 8gtF 93
1f59: 3sAm Q0uw 0i0A 1SAy Xw04 WEkh 21
1f6b: FLWB %!2a 1W19 xv#B i8n# 82c7 61
1f7d: xgbM 50EI Gge5 4i0x 1W&2 Q09F 8a
1f8f: 221x 1%2q 82c7 YeAw 8wtF 1cA6 e3
1fa1: Aebg 1!0y 1SA6 QdCw 1i0A 1SAa e8
1fb3: !hjM 3sAr QcGw 120A 1SAH Qc6w dd
1fc5: 1Z3R 141i 82c! 84Nf kRgw i1hj 9a
```

```
******************************
* Procedura wyswietlajaca    *
* obrazek w rozdzielczosci   *
* 320x256 w 256 kolorach     *
* Wymagany AA-ChipSet        *
*                            *
*  Coded by Robin/W.F.M.H    *
******************************

Exec                =       4
OldOpenLibrary      =       -408

        lea     Picture,a0
        cmp.l   #'FORM',(a0)
        bne     Get_out

        cmp.l   #'ILBM',8(a0)
        bne     Get_out

        lea     Picture+12,a0
        move.l  #'CMAP',d7
        bsr     Wait_for

        bsr     MakePalette

        move.l  #'BODY',d7
        bsr     Wait_for

        lea     DataPic,a1
        move.w  #255,d6
Loop3   bsr     Take_line
        dbf     d6,Loop3

        lea     BitPointers,a0
        moveq   #7,d7
        move.w  #$e0,d0
        move.l  #DataPic,d1
Lip4    move.w  d0,(a0)+
        swap    d1
        move.w  d1,(a0)+
        addq.l  #2,d0
        move.w  d0,(a0)+
        swap    d1
        move.w  d1,(a0)+
        addq.l  #2,d0
        add.l   #40,d1
        dbf     d7,Lip4

        move.l  #copper,$dff080

WaitLMB btst    #6,$bfe001
        bne.s   WaitLMB
Get_out
        lea     Graphname(pc),a1
        move.l  Exec,a6
        jsr     OldOpenLibrary(a6)
        move.l  d0,a0
        lea     $dff000,a6
        move.l  $26(a0),$80(a6)
        clr.w   $88(a6)
        rts

Wait_for
        cmp.l   (a0),d7
        beq.s   Allright
        move.l  4(a0),d3
        addq.l  #8,a0
        add.l   d3,a0
        bra.s   Wait_for
Allright
        addq.l  #8,a0
        rts

;*****Depacking IFF***************
Take_line
        moveq   #0,d7
Loop    moveq   #0,d0
        move.b  (a0)+,d0
        cmp.b   #$80,d0
        beq.s   Zero
        tst.b   d0
        bpl.s   Data
        neg.b   d0
        moveq   #0,d1
```

```
         move.b    (a0)+,d1
         add.l     d0,d7
         addq.l    #1,d7
Loop1    move.b    d1,(a1)+
         dbf       d0,Loop1
         bra.s     Next1
Zero     addq.l    #1,d7
         bra.s     Next1
Data     add.l     d0,d7
         addq.l    #1,d7

Loop2    move.b    (a0)+,(a1)+
         dbf       d0,Loop2

Next1    cmp.l     #40*8,d7
         bcs.s     Loop
         rts

MakePalette
         lea       ColMap,a1
         moveq     #7,d7
         move.w    #$c00,d0
Loop6    move.w    #$106,(a1)+
         move.w    d0,(a1)+

         moveq     #31,d6
         move.w    #$180,d5
         move.l    a0,-(a7)
         bsr       GetNext
         move.l    (a7)+,a0

         eor.w     #$200,d0
         btst      #9,d0
         bne.s     Bump
         add.w     #$2000,d0
Bump
         move.w    #$106,(a1)+
         move.w    d0,(a1)+

         moveq     #31,d6
         move.w    #$180,d5
         bsr       GetNext

         eor.w     #$200,d0
         btst      #9,d0
         bne.s     Bump2
         add.w     #$2000,d0
Bump2    dbf       d7,Loop6
         rts

GetNext  move.w    d5,(a1)+
         bsr       TakeC
         move.w    d1,(a1)+
         addq.w    #2,d5
         dbf       d6,GetNext
         rts

TakeC    moveq     #0,d1
         btst      #9,d0
         bne.s     Lower

         move.b    (a0)+,d1
         and.b     #$f0,d1
         lsl.w     #4,d1
         moveq     #0,d2
         move.b    (a0)+,d2
         and.b     #$f0,d2
         or.w      d2,d1
         move.b    (a0)+,d2
         and.b     #$f0,d2
         lsr.b     #4,d2
         or.b      d2,d1
         rts

Lower    move.b    (a0)+,d1
         and.w     #$f,d1
         lsl.w     #8,d1
         moveq     #0,d2
         move.b    (a0)+,d2
         and.b     #$f,d2
         lsl.b     #4,d2
         or.w      d2,d1
         move.b    (a0)+,d2
         and.b     #$f,d2
         or.b      d2,d1
         rts

Picture incbin     "df0:NazwaRysunku"

         blk.w     4,0

;Etykieta DataPic musi byc pod adresem
;podzielnym przez 8.

DataPic  blk.l     20480,0

Copper   dc.w      $106,$c40
         dc.w      $1fc,$000
         dc.w      $1e4,$000
         dc.w      $100,$000
         dc.w      $102,$000
         dc.w      $104,$000
         dc.w      $096,$020

         dc.w      $08e,$2c81
         dc.w      $090,$2cc1
         dc.w      $092,$0038
         dc.w      $094,$00d0
         dc.w      $0108,7*40-8
         dc.w      $010a,7*40-8
BitPointers
         blk.l     16,0

         dc.w      $1e4,$2100
         dc.w      $1fc,$0003
         dc.w      $100,$0010

ColMap   blk.w     1056,0
         dc.l      $ffffffe

Graphname          dc.b 'graphics.library',0
```

Cześć Kebabie!

Do napisania tego listu zmusił mnie wstrząsający list Dr. Boczka (Nr 10/92) dotyczący dziwnego zachowania programu "DPaint" bez podłączonej do komputera pamięci FAST, a mianowicie program ten... uruchamia się!

Na własnej skórze zetknąłem się z tym problemem jednak po długich i wyczerpujących testach zarówno sprzętu jak i oprogramowania zdołałem odkryć przyczynę.

Na początku przeprowadziłem szczegółowy przegląd komputera. Po zerwaniu plomby gwarancyjnej i po otwarciu obudowy, wewnątrz odkryłem ok 1/5 litra niezidentyfikowanej cieczy. Szybko odparowałem szkodnika kładąc sprzęt na kaloryfer. Niestety, po złożeniu Amigi program dalej się uruchamiał...

Postanowiłem zatem sprawdzić, czy aby przypadkiem program nie posiadał jakiegoś wirusa. Program o nazwie "ButX" nie odkrył niczego. Podejrzewając, że kłamie, wylistowałem sobie zawartość bootblock'u. Natknąłem się tu na dziwną sekwencję "DOS".

Nie czekając ani chwili dłużej przystąpiłem do działania: wyzerowałem bootblock, zapisałem takowy na dysk, wyresetowałem komputer i... eureka - nie uruchomił się. Zrozumiałem, że "dziwna" sekwencja "DOS" była niczym innym, jak kodem wirusa. Sprawdziłem więc resztę moich dyskietek i naprawiłem je. Niestety okazało się, że teraz wszystkie one wymagają pamięci typu FAST.

Z poważaniem

Dr. Boyler

Heja Dr. Boczku!!!

Zdaje się że mam rozwiązanie na Twój problem z A3000T w której to znajduje się 4MB Chip-Ram'u. Wina tkwi w Agnusie 8372B. Twórcy tegoż układu wyposażyli go w buffer dla Blittera i Coppera o pojemności (bagatela!) 2MB. Pocóż tyle? Otóż te 2 mega służć miało profesjonalnym twórcom plazmy, której CopperLista miała akurat tyle zajmować, a Blitter miał po niej "szaleć".

Twórcy poszli jednak na rękę amatorom i wstawili układ blokujący dostęp do tej pamięci. Jednak profesjonalne zwarcie pomiędzy pinami $ff i $fe spowoduje spalenie się stosownego układu i łatwy dostęp do wyżej wymienionej pamięci. W przypadku niepowodzenia i ciągle widniejącego napisu przedstawiającego ok. 2MB graphics memory bierzemy w prawą (lub lewą, to zależy od upodobań) górną kończynę chwytną impulsownik kinetyczny z naprowadzaczem drewnianym i j.....b w Agnusa.

Winno poskutkować. Jednakowoż może okazać się to szkodliwe dla wirusów plikowych i, jak się ostatnio okazuje, bootblock'owych oraz disk-validator'owych, których przecież tyle się roi w samym Agnusie, że o Denisie czy Pauli nie wspomnę.

To było rozwiązanie hardwarowe. Jeżeli boisz się upośledzenia Swojej Amigi to pozostaje Ci tylko możliwość software'owa. Problem stanowią amatorscy programiści z Commodore Ltd. (lub Inc.) Otóż ci amatorzy pisząc procedurę programu LoadWB zapomnieli, że po odwołaniu się do procedury AvailMem z exec.library należy wynik wolnej ilości pamięci, znajdujący się w rejestrze procesora d0, przemnożyć przez 2. Stąd zamiast 4MB graphics mem Masz tylko 2MB.

Poniższa procedura napisana w asemblerze rozwiązuje problem. Oprócz tego (jako skutek uboczny) pozwoli Ci ona również podwoić ilość wolnej pamięci Fast-Ram...

Teraz wszyscy profesjonaliści mogą spać spokojnie.

Z Amigowskim poważaniem dla amatorów i profesjonalną dozą dezaprobaty

Crimol

```
AvailMem = -216
SetFunction = -420

section MemoryAdderPro0,code

Start lea
Start-4(pc),a0
move.l (a0),d7
clr.l (a0)
addq.l #2,d7
rol.l #2,d7

move.l 4.w,a6
move.l a6,a1
move.w #AvailMem,a0
move.l d7,d0
move.l d7,a5
jsr SetFunction(a6)
move.l d0,-4(a5)
moveq #0,d0
rts

section MemoryAdderPro1,data

OldVect dc.b 'PRO!'

NewVector
movem.l a1,-(sp)
move.l OldVect(pc),a1
jsr (a1)
add.l d0,d0
movem.l (sp)+,a1
rts
```

Stukrotne dzięki dla wszystkich Czytelników, którzy zechcieli się podzielić z Dr. Boczkiem swoimi doświadczeniami. Mamy nadzieję, że wasze listy pomogą mu jak najszybciej wspiąć się na wyżyny profesjonalizmu. Niestety droga ta nie jest wcale łatwa, czego dowodzą kolejne listy...

Drogi Kebabie,

to znowu ja (Dr.Boczek)! Być może macie już mnie dosyć, ale tym razem mam tylko małe pytanko. W jednym z profesjonalnych numerów magazynu fanów miesięcznika, zauważyłem piękny napis DFU. Wiedziony nieodpartą chęcią poszerzenia swojej wiedzy, postanowiłem sprawdzić co też może się kryć pod tak profesjonalnie brzmiącym tytułem. I tu olśnienie! DFU to po prostu:

Daten Funk Uebertragung.

Dowiedziałem się również, że te trzy (!) wyrazy oznaczają przenoszenie danych za pomocą sieci telefonicznej. Wszystko jasne! Nie byłoby żadnego problemu, gdybym nie był wścibski. Otóż zacząłem się zastanawiać skąd można wyszukać takie profesjonalne wyrazy? Na przyszłość jak znalazł!

Wiedziony intuicją domyśliłem się od razu, że skoro "polski magazyn zajmuje się profesjonalnie problemami użytkowników Amiga", to muszę sięgnąć po słownik niemiecko-polski. Tu... życie stało się prostsze! Odkryłem amatorskie odpowiedniki w/w haseł. "Daten" to "dane", "Funk" to "radio". Gorzej poszło mi z trzecim wyrazem, ale domyśliłem się, że "Uebertragung" to mniej więcej "przenoszenie". Teraz czas na właściwe pytanie. Jeżeli połączę ze sobą te trzy wyrazy, to od razu mam ochotę się zapytać czy DFU to profesjonalny odpowiednik wspominanego przez was Packet Radio? Proszę wyjaśnijcie moje wątpliwości! W waszych artykułach na temat PR nic nie ma o sieci telefonicznej!

Wasz Dr. Boczek

Cześć Dr.Boczku!

Czy mamy Cię dosyć? Jak na razie jeszcze nie! W końcu dzięki Tobie zarówno nasi Czytelnicy jak i my możemy się dowiedzieć wielu ciekawych rzeczy. Przechodząc jednak do pytania, musimy stwierdzić, że nie znaleźliśmy dotąd takiego hasła (skrótu) jak DFU. Nie wiemy również co ma wspólnego "przenoszenie danych za pomocą sieci telefonicznej" z radiem.

Być może profesjonaliści używają radia podłączonego do sieci telefonicznej...?! W naszym gronie używamy terminu "Komputerowa Telekomunikacja" ew. w skrócie KTK na przesyłanie danych na odległość przy użyciu np. linii telefonicznej. Ale oczywiście nie powinieneś się tym sugerować. Być może jednak Twoja intuicja cię zawiodła i wyjaśnienia tych tajemniczych haseł należałoby szukać w jakimś innym słowniku... W języku niemieckim jest nam znany skrót

DFÜ

oznaczający "Datenfernübertragung" czyli mniej wiecej tyle co "przesyłanie danych na odległość" i to wszystko co na ten temat wiemy. Może i tym razem ktoś z Czytelników będzie mógł Ci pomóc...

Kupon ogłoszeniowy

imię i nazwisko

adres

treść:



Cześć Kebabie!

miałem wam dać spokój, ale czytając jeden z profesjonalnych numerów dowiedziałem się, że autor programu PC-Task zapowiedział wydanie emulatora AT. Słyszałem, że KEBAB ma z nim bezpośredni kontakt. Moje pytanie: Czy macie już ten emulator? Błagam przyślijcie mi, bo nie mam już czym dręczyć pecetowców w mojej klasie!

Dr. Boczek

Drogi Dr.Boczku,

nieprawdą jest, że KEBAB ma bezpośredni kontakt z Chrisem Hames'em odpowiedzialnym za popełnienie PC-Task'a (skrót myślowy od: popełnienie tej zbrodni na pecetowcach, jaką było napisanie PC-Task'a). Prawdą natomiast jest, że nasz kolega redakcyjny miał z nim sporo łączności zarówno telefonicznych jak i pocztą elektroniczną.

Niestety potraktował nas jak przystało traktować amatorów... Powiedział, że nie zamierza aktualnie programować emulacji AT, gdyż zajęty jest w tej chwili rozszerzeniem emulacji graficznych PC-Task'a o niektóre tryby EGA/VGA.

Jeżeli chodzi o przesłanie programu, to jak otrzymamy nową jego wersję, będziemy mogli przesłać Ci tylko wersję demonstracyjną. W przeciwnym przypadku okradalibyśmy bardzo zasłużonego dla Amigi programistę...